HELSINKI UNIVERSITY OF TECHNOLOGY
Faculty of Information Technology
Department of Computer Science and Engineering
Telecommunications Software and Multimedia Laboratory

Kimmo Karhu

Domain analysis of location-based and context-aware mobile services

Master's Thesis
November 2002

Supervisor Professor Tapio Takala
Instructor Professor Norisha Komoda

HELSINKI UNIVERSITY OF TECHNOLOGY Department of Computer Science and Engineering	ABSTRACT OF MASTER'S THESIS

Author:	Kimmo Karhu
Title:	Domain analysis of location-based and context-aware mobile services
Translation in Finnish:	Sovellusalue-analyysi paikkapohjaisista ja käyttökontekstin mukaan mukautuvista langattomista palveluista
Language:	English **Pages:** 6+67
Date:	November 6th, 2002
Field of study:	Interactive digital media
Supervisor:	Professor Tapio Takala
Instructor:	Professor Norisha Komoda

Abstract:

Context-aware computing has already been researched for a decade but it has never really found its way into commercial use. Lately, the growth in the use of mobile services and the introduction of new technologies has raised interest again. In this paper, the domain of location-based and context-aware mobile services is analysed and a domain model using Unified Modelling Language (UML) is presented. The model facilitates the development of domain-specific software framework. The implementation of the framework has been left for future work; in this paper, only some design guidelines for the architecture are suggested. This work also serves as a good example of how UML can be used for the domain modelling in general and thus the results can also be applied to other domains.

Keywords: mobile service, location-based, context-aware, domain analysis, UML

TEKNILLINEN KORKEAKOULU Tietotekniikan osasto	DIPLOMITYÖN TIIVISTELMÄ

Tekijä:	Kimmo Karhu
Työn nimi:	Domain analysis of location-based and context-aware mobile services
Suomenkielinen nimi:	Sovellusalue-analyysi paikkapohjaisista ja käyttökontekstin mukaan mukautuvista langattomista palveluista
Kieli:	Englanti **Sivumäärä:** 6+67
Päivämäärä:	6. marraskuuta, 2002
Pääaine:	Vuorovaikutteinen digitaalinen media
Työn valvoja:	Professor Tapio Takala
Työn ohjaaja:	Professor Norisha Komoda

Tiivistelmä:

Käyttökontekstin mukaan mukautuvia sovelluksia on kehitetty jo vuosikymmenen ajan, mutta niistä ei ole vieläkään syntynyt suurempia kaupallisia menestyksiä. Viime aikojen kehitys langattomissa teknologioissa ja kasvu palveluiden käytössä ovat uudestaan herättäneet kiinnostuksen sovellusalueeseen. Tässä työssä tehdään sovellusalueanalyysi paikkapohjaisista ja käyttökontekstin mukaan mukautuvista langattomista palveluista ja esitellään malli sovellusalueesta käyttäen UML mallinnuskieltä. Malli on tarkoitettu helpottamaan sovellusaluekohtaisen ohjelmistokehyksen kehittämistä. Itse toteutus jätetään tehtäväksi tulevaisuudessa ja siten tässä työssä ehdotetaan ainoastaan yleisiä suunnitteluperiaatteita toteutettavalle arkkitehtuurille. Tämä työ voidaan myös nähdä yleisenä esimerkkinä UML:n käytöstä sovellusalueanalyysissä ja näinollen tuloksia voidaan käyttää mallitettaessa muitakin sovellusalueita.

Avainsanat: langaton palvelu, paikkapohjainen, käyttökonteksti, sovellusalueanalyysi, UML

Preface

This work was done for Codetoys Ltd. during 2001-2002.

I would like to thank Professor Tapio Takala for supervising the work and for giving valuable instructions. This helped to organize the paper and finally complete it for returning.

This work was mainly done in Japan at Osaka University. I wish to thank the personnel and students of the Komoda lab for their support. I sincerely want to thank my instructor Professor Norisha Komoda for helping to define the problem field, guiding the work and for teaching me how to write a research paper. I would also like to thank Codetoys personnel in Japan for their continuous support.

This work was financed by Codetoys. I wish to thank the personnel of Codetoys for the possibility to concentrate on the research work and to finally write the thesis. Especially I would like to thank my colleague Mikko Rusama for valuable lunch discussions.

Finally, I want to thank my dear Hanna and my family for their support and encouragement. Special thanks to my parents for providing a quiet and peaceful place to work.

Espoo

November 6th, 2002

Kimmo Karhu

Table of Contents

List of Figures

List of Tables

1 Introduction

The rapid development in two areas, handheld devices and wireless communications, has characterized the information technology industry during recent years. The industry has seen a remarkable boom and growth of users fulfilling some of the promises of tomorrow's mobile world. The keywords of that world are "anytime, anywhere". To help users to manage in this complex and changing world, applications have to be more intelligent and aware of the things happening around them. In other words, they need to be context-aware.

The demand for personalized and ubiquitous computing can be seen in the research world. During recent years there have been more and more papers published in the field of context-awareness and location-based services. Context-awareness in general has already been researched for a decade and also location-based services and techniques related to it are very well understood. However, despite the research activity, so far context-awareness has not found its way into commercial use. Most of the context-aware applications and services of today are either prototypes being developed in the universities or simple location-based services. Lots of work remains to be done before this technology is part of our everyday life.

However, lately two new promising technologies for short-range networking, Bluetooth and Wireless LAN, have emerged providing a technology to implement location-based services. Adding context-awareness to this opens interesting possibilities. For example, in a sport event at a stadium there could be personalized services available for different team supporters located in different parts of the stadium. In a museum, a visitor could get personal guidance and additional information through his mobile terminal as he walks around in different locations. So far this specific application domain has not been widely studied, thus there is a motivation for this research.

In this paper, we will examine this domain and find out what it is about. We will go through example applications and produce a general model for the domain. At the same time, we will collect and introduce some new terminology to be used. These together will hopefully help software developers working in this field. Understanding the domain and having a model in hand make it much easier to start the development work.

1.1 Structure of the paper

The structure of the paper is as follows. The next sections in this chapter introduce the terminology, define the research problem and present the previous work. Chapter 2 defines the criteria that will be used to evaluate the proposed solution. Chapter 3 contains the domain analysis, the main result of this work. The sections for that chapter are as follows. Section 3.1 introduces methods that were used. Section 3.2 presents a generic model, the big picture. Section 3.3 describes the four example applications that were the main input for the analysis. In this section, a use case analysis is carried out for each application. Section 3.4 defines the boundaries for the domain and Section 3.5 presents the domain model that was produced as a result of the analysis. Finally, some architecture modelling is done in Section 3.6 and some design guidelines are suggested for a domain-specific framework. Chapter 4 contains the evaluation of the solution where the proposed model is evaluated against two applications. At the last, Chapter 5 concludes this paper and lists some future work.

1.2 Definition of terms

1.2.1 Interactive service

There are many definitions for service and the term is used for various meanings. In this paper service is most often used for a software application that resides somewhere on the server and is used by the client (Software Technology Review, 2002). Client means both the user and the mobile terminal device he is using. An interactive service is a service where there is some interaction between the user and the service. Some of the different service types are:

- Entertainment services: WAP&SMS games, such as Who Wants to Be a Millionaire
- Information services: Yellow pages, timetables
- Surveys: Mobile marketing research surveys

In addition, combinations of these can exist. For example, a combination of a game and a survey could be used for sales promotion.

1.2.2 Short-range networking

Short-range networking term is used in this paper to refer to wireless networking standards for short distances such as Bluetooth and Wireless LAN (WLAN). Both of these technologies use radio technology for the network link. WLAN is an IEEE specification (sometimes referred as 802.11x), which has been seen as the wireless replacement for the traditional wired LANs. Bluetooth instead is an industry driven standard that was originally meant to be used for very short distances and for cases where low power consumption is needed. Some typical scenarios are the communication between a mobile phone and a laptop or between a PDA and a mobile phone. Nowadays, both standards do more or less the same thing and it remains to be seen which one becomes the de facto standard.

1.2.3 Context-awareness

Context-awareness has already been researched for more than a decade and many people have proposed various definitions for "context". Dey et Abowd (1999) define context as "any information that can be used to characterize the situation of an entity. An entity is a person, place, or object that is considered relevant to the interaction between a user and an application, including the user and applications themselves". Whereas Chen and Kotz (2000) define it as "the set of environmental states and settings that either determines an application's behaviour or in which an application event occurs and is interesting to the user". Since it is very loosely used expression, it is difficult to come up with one definition that suits all applications. Nevertheless, these two definitions give a good enough idea that can then be further understood by categorizing different contexts and thinking how context information can be exploited in applications.

Schilit et al (1994) introduced a categorization of contexts, which was further extended by Chen and Kotz (2000). They proposed to add a time category thus having four different categories as follows:

Computing context

Computing context comprises all device-related contexts, such as network connectivity, communication bandwidth and terminal type.

Physical context

Physical context describes the user's physical environment. These contexts include lighting, noise levels and temperature.

Time context

This contains contexts such as time of day, week, month and season of the year.

User context

User context is perhaps the most important context. It enables services that are personalized according to the user's profile and preferences. Contexts in this category include identity and demographic information about the user, the user's interests and preferences (user profile), usage history and location.

Having defined the context, we will next show how the applications can exploit it. Park et al. (2002) identify two ways to do it: contextual information and context-triggered actions.

Contextual information

Applications that are aware of the current executing context alter their presentation according to the context. For example, depending on the user's location, a context-aware museum information system would only show information about nearby exhibits. Another example is a WAP service that alters presentation according to the phone model that the user is using. Exploiting contextual information affects both the content and the way in which it is presented.

Context-triggered actions

A change in the context causes an application to trigger an action or pull some content for the user. For example, an entertainment service in the football stadium may trigger a mobile game when a goal is made. A tourist information system could warn the visitor if a bus connection is cancelled.

1.2.4 Location-based service (LBS)

"Location-based services are services that exploit knowledge about where the user is located" (Whatis?com, 2002). Location information has been the most used and the most important context in the mobile applications to date. According to Park et al (2002), LBS enabled services have been rolled out starting already January 2000 from Japan (DoKoNavi) and Europe following later the same year. It has been estimated that by 2006 even 60% of the mobile applications will use location information. Park et al (2002) claims that LBS enabled services will be an important driving force for the mobile Internet and can eventually evolve to context-aware services. As mentioned before, user's location is just one context among the others. However, there are reasons why it is very important. Firstly, it is the most obvious context for mobile services due to nature of mobility. Secondly, mobile networks provide inexpensive cell-based location estimation technology, which does not require any extra devices such as Global Positioning System (GPS) sensor. Accuracy is worse in cell-based technologies but good enough for most of the applications. Moreover, the largest companies in the industry have set up the Location Inter-operability Forum (LIF) and agreed to produce a common standard (LIF TS 101 Specification, 2002) for querying location information from a wireless network. This work significantly facilitates the development of LBS enabled services.

There are many different positioning technologies available to estimate user's location. The most important ones are:

1. Global Positioning System (GPS)
2. Cell based location method used in the GSM networks
3. Bluetooth and WLAN have methods that use the signal strength and access point locations

1.2.5 Software framework

A software framework is an "extendable set of objects for related functions" (Larman, 2002). Froechlich et al (1998) define it more specifically by saying that "A framework doesn't cover all of the functionality required by a particular domain, but instead abstracts the common functionality required by many applications, incorporating it into common design and leaving the variable functionality to be filled in by the framework user".

1.2.6 Domain analysis concepts

Major part of this thesis is about domain analysis. To understand the analysis that will be carried out later it is essential to comprehend the basic domain analysis terminology first. Kang et al (1990) have defined the terminology as follows:

Application: A system that provides a set of general services for solving some type of user problem.

Context: The circumstances, situation, or environment in which a particular system exists.

Domain (also called application domain): A set of current and future applications, which share a set of common capabilities and data.

Domain analysis: The process of identifying, collecting, organizing, and representing the relevant information in a domain based on the study of existing systems and their development histories, knowledge captured from domain experts, underlying theory, and emerging technology within the domain.

Domain engineering: An encompassing process which includes domain analysis and the subsequent construction of components, methods, and tools that address the problems of system/subsystem development through the application of the domain analysis products.

Domain model: A definition of the functions, objects, data, and relationships in a domain.

Feature: A prominent or distinctive user-visible aspect, quality, or characteristic of a software system or systems.

Software architecture: The high-level packaging structure of functions and data, their interfaces and control, to support the implementation of applications in a domain.

Software reuse: The process of implementing new software systems using existing software information.

Reusable component: A software component (including requirements, designs, code, test data, etc.) designed and implemented for the specific purpose of being reused.

User: Either a person or an application that operates a system in order to perform a task.

1.3 Problem statement

The purpose of this section is to define and explain the problem to be solved in this work. Section 1.3.1 presents the objectives of this thesis and Section 1.3.2 defines the scope for the research.

1.3.1 Objective

The objective of this thesis is to analyse and produce a model for the domain of location-based and context-aware mobile services.

The first objective of this thesis is to analyse the problem domain. Analysis is required to identify common features among the applications that are reused in a framework. As the result of the analysis, a domain model will be presented that captures requirements relevant to the system design. It includes both properties common to all applications, and properties characteristic of individual applications in the domain (Baum et al, 2000). Figure 1 illustrates the domain analysis process.

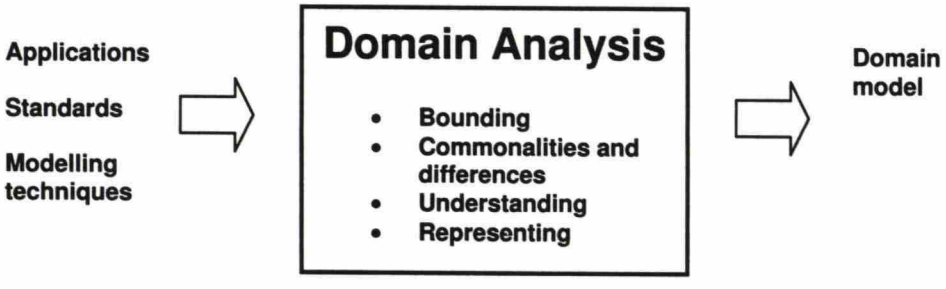


Figure 1. Domain analysis process

One purpose of this thesis is to describe how the domain relates to other domains and the general field. The intention is to reveal the difference between the domains and to show that a separate framework should be developed.

The second objective is to propose some design guidelines for a domain-specific software framework. Based on the findings from the analysis, the most important parts of the software framework will be identified.

1.3.2 Scope

This paper concentrates on interactive wireless services. Only wireless applications and client devices are explored, though many of the services could also be implemented using fixed line networks. Moreover, no certain type of interactive service is excluded. Example applications used as an input will cover many kinds of interactive services such as entertainment, infotainment and education.

The model should be independent of the underlying network technology. It should be possible to deploy services using most of the wireless technologies including new short-range wireless standards such as Bluetooth and Wireless LAN. Special attention is given to Bluetooth as it is widely recognized as the future standard in the industry.

When utilizing context-awareness, the focus is on the mobile environment. This paper will look into the contexts that can be utilized for mobile terminals. Less attention is given to the physical and computing contexts, such as temperature, network connectivity and characteristics of the access device. Instead, this thesis concentrates on user contexts, which are, for example, location, user profile and access history. This paper concentrates on context-awareness in general and implementation details are omitted.

There are already many platforms and software frameworks for developing interactive wireless services, but not that many exist for creating context-aware and location-based services. This paper will identify the missing parts for this specific domain. Implementation of the framework is outside the scope of this thesis but some design guidelines for the architecture are proposed.

This thesis is not a survey of different technologies in the domain but rather an analysis of the domain from a software point of view. Relevant technologies are introduced but they are not studied in detail.

1.4 Referenced work

In this section, the previous work that has been done and is referenced in this paper is introduced. A complete list of all references can be found from the end of this paper, here only the most important works are introduced.

As mentioned earlier, context-awareness has already been studied for a decade. In this work, we will mainly reference to the following papers when speaking about context-awareness in general:

- Context-aware computing applications (Schilit et al, 1994)
- Towards a Better Understanding of context and context-awareness (Dey and Abowd, 1999)
- A Survey of Context-Aware Mobile Computing Research (Chen and Kotz, 2000)
- Location Based Services for Context Awareness – Moving from GSM to UMTS (Park et al, 2002)

The first two papers deal with the terms and definitions related to context-awareness. Third paper is a comprehensive survey of the research concerning context-awareness. It lists both the research efforts and the applications that have been implemented in this field. Fourth paper studies context-awareness from the location-based services point of view. It reveals the close connection between these two and shows how this relates to UMTS.

The other important works referenced are:

- Feature Oriented Domain Analysis (Kang et al, 1990)
- Applying UML and Patterns: an introduction to object-oriented analysis and design and the Unified Process (Larman, 2002)

Both of these works study domain analysis. The former has a very analytical and structured approach to the problem whereas the latter is more practical. One of the strengths of the Larman's book is that it is very up-to-date following the latest trends in the software industry: Unified Modelling Language (UML) and Unified Process (UP). The analysis carried out in this work and especially the presentation is mainly based on the Larman's book. We will also follow the use case driven methodology presented in the book. The former paper, Feature Oriented Domain Analysis, will be used as a general reference of

domain analysis method as well as a guideline to what are the most important outputs from it. Finally, it is worth of mentioning the existing applications in the domain, which were the main source of information for the analysis and evaluation. They are:

- Developing a Context-aware Electronic Tourist Guide: Some Issues and Experiences (Cheverst et al, 2000)
- The development of information service system using Bluetooth in an exhibition hall (Kishimoto et al, 2001)
- Cyberguide: A Mobile Context-Aware Tour Guide (Abowd et al, 1997)
- Pango Networks' Proximity Platform (Proximity Platform, 2002)
- The Conference Assistant: Combining Context-Awareness with Wearable Computing (Dey et al, 1999)
- FieldNote: a Handheld Information System for the Field (Ryan et Pascoe, 1999)

The first four of these applications were used as an input for the analysis. They will be introduced and explored in detail using use case analysis in Section 3.3. Latter two are used for evaluation and are studied in Chapter 4.

2 Criteria

This chapter presents the criteria that are used to evaluate the quality of the proposed model. The evaluation is done in Chapter 4 where two example applications are used and it is studied how well each of these criterions can be fulfilled.

Criterion 1. Generality and Applicability

The proposed model should be applicable to different applications and concepts within the domain. Abstractions should be flexible so that most of the variations in the domain can be handled. The elements contained in the model should be general so that the corresponding elements from different applications can be fitted into it. In addition, from the technology point of view the proposed model should be abstract enough so that it can be adapted to most of the existing technologies used in the domain. It is also required that the model should not be outdated in the near future, but instead new technologies and concepts could be abstracted and adapted into it. Some examples of these are: network (support for most of the wireless network standards), platform (independency of the underlying application and server platforms) and context (support for the existing and future contexts).

Criterion 2. Completeness

The model should cover most of the applications, concepts and main technologies within the domain. It is not enough to cover only few applications and technologies because the proposed model is intended to be used for developing a domain-specific software framework and not an application specific software framework. The model should also be complete in a sense that it covers all aspects of an application: features and functionality, entities and their relationships, application context and relations with other software systems and technologies.

Criterion 3. Understandability

The model should be easy to use, understand and facilitate. A person without any prior knowledge of context-awareness and location-based services should be able to comprehend and use the model. The vocabulary and abstractions that are used should be such that it is easy to match the corresponding words and concepts for a specific application.

Criterion 4. Consistency

The work should be consistent with the previous work and existing standards. The definitions and standards existing in the industry and the research world should be used whenever possible.

3 Domain analysis

A successful software reuse requires systematic discovery of commonality across the related applications (Kang et al, 1990). Domain analysis is one method that can be applied and was chosen for this paper. Domain analysis consists of the methods and the outputs that help to produce reusable software components that can be used to create new applications in the domain. Analysis has various phases including defining the scope of the domain, domain model creation and architecture modelling. All of these are later analysed in more detail.

This chapter presents the analysis for the domain that was preliminary scoped in the problem statement. As a systematic method for the analysis, Feature-Oriented Domain Analysis (FODA) is applied (Kang et al, 1990). There are also many other methods available but FODA was selected for the following reasons:

- It is general enough
- There are good documentation and examples available
- It is flexible; neither specific modelling tool nor programming language is required

Due to characteristics of the analysed domain the outputs of the method had to be modified and some of them were not applicable at all. Because the work has been published quite a while ago the modelling examples suggested in the paper are already outdated. For this reason, FODA was used more as a guideline for what should be done and actual models and diagrams were drawn using UML and following the way presented by Larman (2002).

This chapter consists of six sections. Section 3.1 introduces the UML and FODA method. Section 3.2 presents a generic model of the domain. Section 3.3 describes the applications that were used as an input for the analysis. Section 3.4 specifies more precisely the domain and its boundaries to the other domains. The actual modelling is done in Section 3.5 and Section 3.6 presents some architecture guidelines for a domain-specific software framework.

3.1 Introduction to UML and FODA

"The Unified Modelling Language (UML) is a graphical language for visualizing, specifying, constructing, and documenting the artifacts of a software-intensive system. The UML offers a standard way to write a system's blueprints, including conceptual things such as business processes and system functions as well as concrete things such as programming language statements, database schemas, and reusable software components" (OMG Unified Modelling Language Specification, 2002).

Feature-Oriented Domain Analysis has been developed in Carnegie Mellon University. It is a method for performing domain analysis by focusing on the user-visible features. The primary goal of the method is the development of generic and reusable domain components. The basic modelling concepts are abstraction and refinement. Generic components are created by collecting features from various applications in the domain and abstracting out "factors" that make one application different from the other. It is important to understand what differentiates applications as well as the commonalities between them. (Kang et al, 1990)

The method that we will use to identify features is use case analysis. It is a widely used mechanism to record requirements and discover functional features (Larman, 2002). In its simplest form it is simply writing stories of using a system, i.e. use cases. More details of the used method can be found from Section 3.3.

As an input for the analysis various sources are used:

- **Textbooks.** Good source of general domain knowledge, underlying theories, methods and techniques.
- **Standards.** Represents standard reference model for the domain.
- **Existing Applications.** The most important source of user-visible features. At least three applications should be used.
- **Domain Experts.** Can provide contextual information and rational understanding unavailable from elsewhere.

Feature-Oriented Domain Analysis introduces methods to perform the analysis and describes the products and processes for various phases. Method consists of three phases each having various products. These are illustrated in Figure 2.

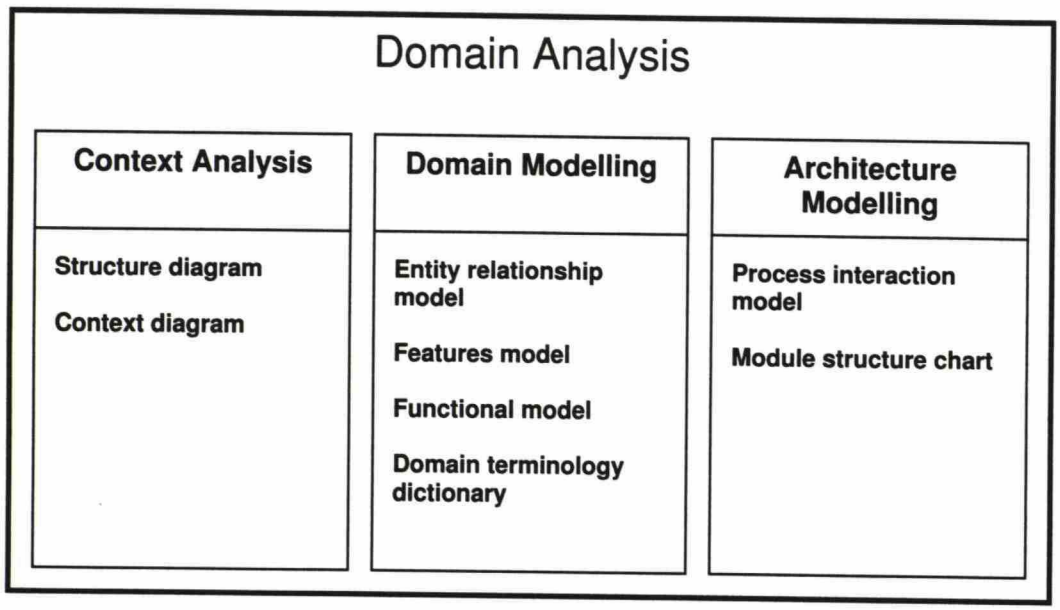


Figure 2. Phases and Products of FODA

Context Analysis

The purpose of the context analysis is to scope the domain and to define the interfaces to other software systems. Structure diagram defines the scope of the domain and locates it according to other domains. Context diagram defines the interfaces and data flows between the different software systems.

Domain Modelling

In the modelling phase, requirements and problems addressed by the applications of the domain are collected. Entity relationship model describes the entities in the domain and their relationships. Feature model collects the generalized features from the analysed applications. These two models provide the static view of the domain. To capture also the dynamic aspect a functional model is needed. It captures the behaviour and functionality of the applications in the domain.

Architecture Modelling

Architecture modelling represents the solutions in a general level for the problems addressed in the modelling phase. Outputs for this phase present a design reference model, which can then be used as a basis for a detailed design and component construction.

3.2 Generic model

The evolution of interactive wireless services can be seen as the next step from the Internet. Traditionally, the Internet has been accessed using wired networks, such as modem connection or local area network. Thanks to the development of wireless technologies, nowadays it is possible to use these services on a mobile device. In this new environment the personal computer has been replaced with a device capable of wireless communications such as a mobile phone or a handheld PDA. Otherwise, the model is quite the same. As we can see from the Figure 3, three entities can be identified: client, server and network. Generally this is called the client-server model.

Client

Client is the terminal device that user is using to access the service. It sends user's requests to the server and shows the response from the server. In case of the traditional Internet, typically a personal computer equipped with a browser software is used. Large screen and powerful processing power enable complex and graphically rich services. Whereas wireless services are accessed using a smaller and more limited mobile device, which of course sets restrictions on what kind of services can be used.

Network

In a distributed environment, a network is needed between the client and the server. Typically, the route to the server consists of various different networks connected to each other. For example, if services are accessed using a mobile phone, a mobile network such as GSM is needed to connect to the wired network and the Internet. In case of Bluetooth, network consists of wireless radio link to the access point and the local area network (LAN) from that on.

Server

Server provides services and applications for clients. Typically, on the server side a separate application server and a database are deployed as shown in the Figure 3. This kind of architecture, where a separate middle tier server is between the user interface and data management component, is called a three-tier architecture (Software Technology Review, 2002).

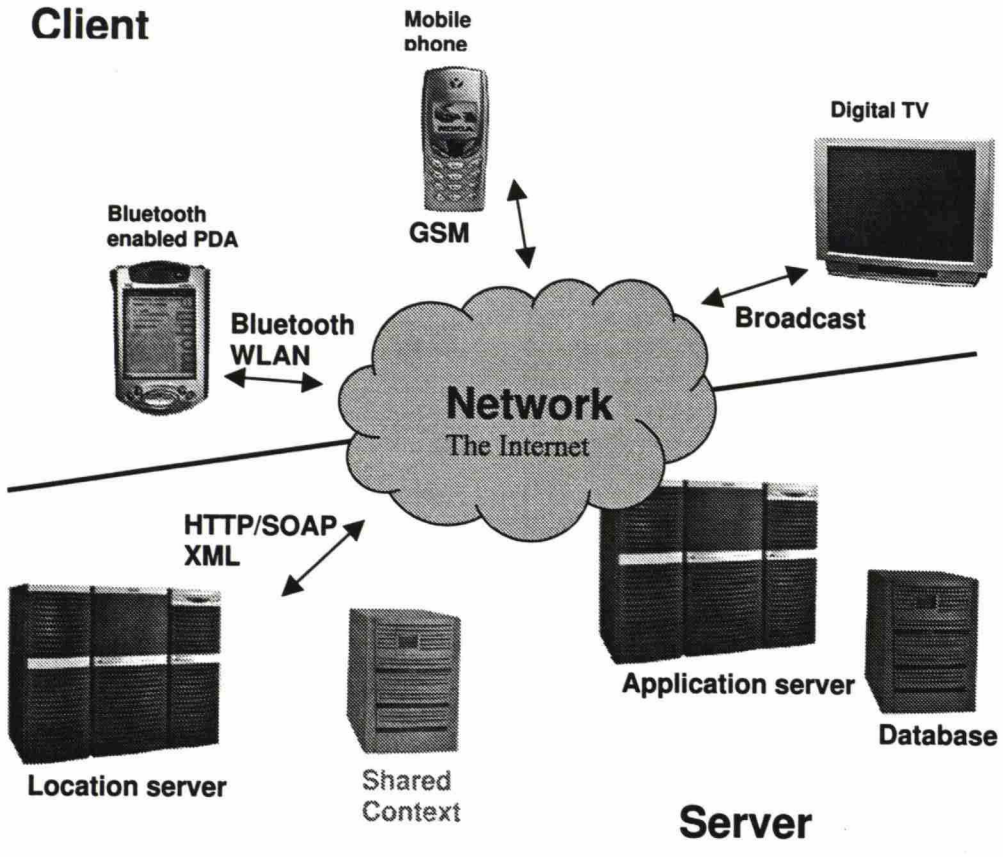


Figure 3. Client-server architecture behind wireless services

The figure also shows some other servers such as a location server and a shared context. In case of location-based services there is usually a location server providing client's location information. Typically, location server provides some API (such as XML messages over HTTP) for querying client's location. This data flow will be further analysed in Section 3.4.2. "Shared Context" in the figure enables new type of services where users' personal preferences and profile are stored in one place in the network and shared among multiple services. This is seen very important and has been addressed in the W3C working groups (Manes, 2001). One example is Microsoft's .NET services model, which provides global authentication services and access to the user's personal data (Microsoft .NET Services, 2002).

3.3 Applications

In this section, the four example applications are described that will be used as an input for the analysis. As this domain is quite new, there were only few applications available to be analysed. For this reason, one envisioned application is also used.

Use case analysis that is used to describe the example applications consists of two phases. Firstly, primary and supporting actors and their goals are identified by examining features of each application. The complete list of features of each application can be found from Appendix C. Secondly, brief use case stories describing applications' functionality are written for each goal. The presentation follows the way suggested by Cockburn (2001) and modified by Larman (2002). In addition to the stories, an example of a fully dressed use case is presented for the "use service" -use case of the stadium entertainment system.

3.3.1 Exhibition tour

3.3.1.1 Description

Exhibition tour application is an information system providing contextual information and guidance for the visitor as he walks around in an exhibition hall. It also provides tools for the manager to manage exhibition layout and to analyse visitor flow. Figure 4 shows a system overview for the application. This application is been developed in the department of information systems engineering at Osaka University. A more complete description of the application can be found from the paper by Kishimoto et al (2002).

3.3.1.2 Actor-Goal list

Table 1. Actor-Goal list for exhibition tour application

Actor	*Goals*
Visitor	*Specify own tour of interest or select a recommended tour* *Get tailored guidance according to the tour* *Get information about nearby and interesting exhibits*
System Administrator	*Manage exhibition layout* *Analyse visitor flow*
System	*Log visitor activity* *Cache content*

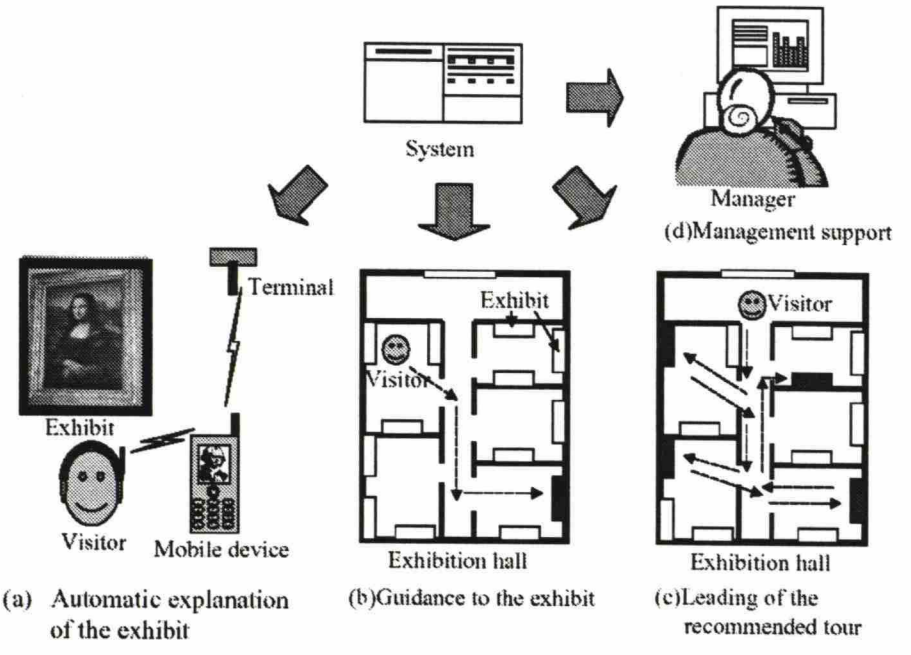


Figure 4. System overview of the exhibition tour application (Kishimoto et al, 2002)

3.3.1.3 Use case stories

Specify Tour

Visitor arrives to the reception of the exhibition hall. The receptionist gives him a mobile terminal and instructs him on how to use it. Visitor specifies his own tour according to his interests or selects a pre-defined recommended tour from the menu. Visitor starts the tour.

Get Guidance

When Visitor walks around in exhibition hall, System guides him according to the tour he has selected. For example, System shows on the screen a map or tells directions to find the next exhibit. System automatically detects Visitor's location and updates that information on the screen.

Get Information

When Visitor arrives to an exhibit System automatically shows the information on the screen and plays a tape describing the exhibit. Visitor can use his terminal to browse the information regarding the exhibit.

Manage Layout

Before the exhibit is opened for Visitors, Manager uses management software to specify the exhibition layout and the recommended tours. Manager uses the same tool to keep the information up-to-date when something is added or removed from the exhibition.

Analyse Flow

During the exhibition Manager uses management software to monitor and analyse the visitor flow. Manager can see from the analysis if some exhibits cause bottlenecks and may redesign the layout.

Log Activity

When Visitor moves around in exhibition hall and explores the exhibits system logs all the activity and the time that Visitor spends in different parts of the hall. This information is used to generate visitor flow reports for Manager.

Cache Content

When Visitor explores an exhibit, System caches on the background information that may be shown next. The exhibits that are cached locally into the terminal are selected based on the selected tour and Visitor's location. Caching is done to improve system's response time.

3.3.2 Stadium entertainment system

3.3.2.1 Description

Stadium entertainment system provides timely and personalized information for the spectators at stadium. In a sport event there could be tailored services available for different team supporters that are located in different parts of the stadium. Special promotions, advertisements and competitions could be triggered during the different times of the match: before the match, halftime, when a goal is made or after the match. Naturally, the event organizer has a tool to map services to the different contexts. This application is completely envisioned and based only on the discussions in the company.

3.3.2.2 Actor-Goal list

Table 2. Actor-Goal list for stadium entertainment system

Actor	*Goals*
Spectator	*Discover interesting and relevant services* *Use context-aware services*
Organizer	*Setup services* *Update services real-time*
System	*Activate context-triggered services*

3.3.2.3 Use case stories

Discover Services

When Spectator moves around in stadium, System shows him a list of services relevant in that particular area. Spectator selects a service that interests him. At any time if a special pre-configured change in the context occurs System may automatically trigger a service.

Use Service

Spectator has discovered and selected a service. System provides the service and tailors content and presentation according to the current context. A fully dressed form of this use can be seen in the Table 3. Table format follows the way suggested by Cockburn (2002).

Setup Services

Before the event, Organizer configures different services to be available in different parts of the stadium and in different contexts. Firstly, Organizer defines spaces that map to certain physical areas and then he maps services to these spaces. Organizer can also map services to other contexts such as time and event.

Update services real-time

When there is a change in context (for example, when a goal is made), Organizer can use a tool to update information into the system.

Table 3. Use Service use case in a fully dressed form

USE CASE	Use Service	
Goal in Context	Spectator has selected a service and wants to use it.	
Primary Actor	Spectator	
Stakeholders and Interests	Spectator: Wants interesting, entertaining and personalized content. Event Organizer: Wants to keep spectators satisfied by providing timely and personalized content. System: Supporting actor that provides the service.	
Preconditions	Spectator has selected a service.	
Success End Condition	Spectator has been entertained/informed.	
Trigger	Spectator selects a service.	
Main Success Scenario	**Step**	**Action**
	1	Spectator selects a service. → Request
	2	System replies and sends an opening page of selected service. → Response
	3	Spectator selects an action in the service. → Request
	4	System checks active context and tailors content and presentation according to it.
	5	System returns the next page. → Response
	Spectator repeats steps 3-5 until done	
	6	Spectator exits from the service. → Request
	7	System returns to service menu. → Response
Extensions	**Step**	**Branching Action**
	*a	At any time system fails to response: 1a. Spectator tries again. 1b. Spectator disconnects from the service
	1a	Service not available: 1a. Spectator tries another service. 1b. Spectator disconnects.
Special Requirements		
Technology and Data Variations List		
Open Issues		

3.3.3 Tourist Information System

3.3.3.1 Description

Wireless, context-aware tourist information systems overcome many of the limitations of the traditional information services available to the tourists. For example, group-based tours are inflexible with fixed time schedules and are typically made to satisfy the needs of the majority rather than an individual. The guidebooks are another good example of limitation as they get very soon out-of-date and require regular updates. Electronic tourist guides can be kept up-to-date more easily and information given to the tourist can be tailored to his personal needs and location. Figure 5 shows an example screen where some of the features of an electronic tourist guide are listed. There have been many projects going on in this field. The following text is based on two projects: an electronic context-aware tourist guide being developed in Lancaster University (Cheverst, 2000) and Cyberguide that was developed in Georgia Institute of Technology (Abowd et al 1996).

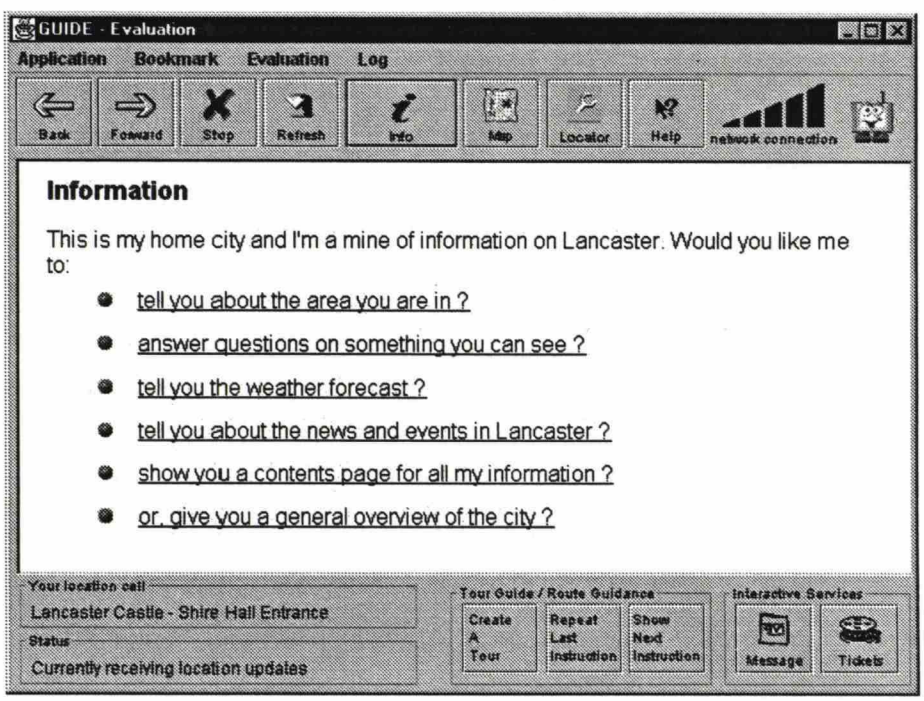


Figure 5. Example screen of electronic tourist information system (Cheverst, 2000)

3.3.3.2 Actor-Goal list

Table 4. Actor-Goal list for tourist information system

Actor	*Goals*
Tourist	*Specify own tour or select a recommended tour* *Get guidance* *Get information about nearby attractions* *Be informed of activities around* *Be informed of changes in schedules etc.* *Communicate with TIC*
System Administrator	*Manage area layout* *Analyse visitor flow*
System	*Log visitor activity* *Cache content*

3.3.3.3 Use case stories

Specify Tour

Visitor arrives to the local Tourist Information Center (TIC). A staff member gives him a mobile terminal and instructs Visitor on how to use it. Visitor may explore the information available in the TIC and discuss with the TIC personnel to find out things that interest him. When finished Visitor can specify his own tour or select a pre-defined recommended tour using his terminal. Visitor starts the tour.

Get Guidance

When Visitor walks around in the area, System guides him according to the tour he has selected. For example, System shows a map on the screen and tells directions to get to the destination. System automatically detects Visitor's location and updates it to the screen. When Visitor leaves an attraction, System tells him the route to the next destination.

Get Information

When Visitor arrives to an attraction terminal automatically presents information about it. Information may be, for example, textual descriptions or audio and video clips. At any point of his tour Visitor may browse general information about the area. This may include information about the city, time schedules of public transportation, opening and closing times, weather forecast, etc.

Be informed

At all times during his visit, Visitor is informed of interesting and important issues. For example, System may alert visitor if he is behind the planned time schedule and may miss some attraction because of the closing time. Visitor may also be informed of a special offer in a shop or a café that he is passing by. Moreover, if time schedules of public transportation or opening and closing times of attractions are changed, it may be communicated to the Visitor.

Communicate with TIC

Any time during his tour Visitor can communicate with TIC using his access terminal. He may ask a specific question from the member of staff or use some other services such as book an accommodation.

Manage Layout

Manager uses management software to setup area layout. He can map information and services to different locations and specify routes and schedules for recommended tours. Manager uses the same tool to keep the information up-to-date when something is changed in the area.

Analyse Flow

Manager uses management software to monitor and analyse the visitor flow. He can identifiy crowded locations and redesign the area to avoid bottlenecks.

Log Activity

When Visitor moves around the area, System logs all activity and the time that Visitor spends in different locations. This information is used to generate visitor flow reports for the Manager.

Cache Content

System caches part of the content to the terminal in beforehand. Based on the selected tour and Visitor's location System can cache information about the next attractions. Caching is done to improve system's response time and to cover places where network is not available.

3.3.4 Proximity platform

3.3.4.1 Description

Pango's proximity platform is a technology that allows third parties to create, deploy and manage personalized location-based services using short-range wireless networks (Pango, 2001). Their model is based on an idea of spaces & services. Services are mapped to spaces and are triggered in user's presence. Figure 6 shows an example of services deployed on a train station. In a similar way, spaces & services model could be applied to other areas such as a football stadium, a hotel, an airport or a shopping mall. Their model allows highly relevant and personalized content to be delivered to the users.

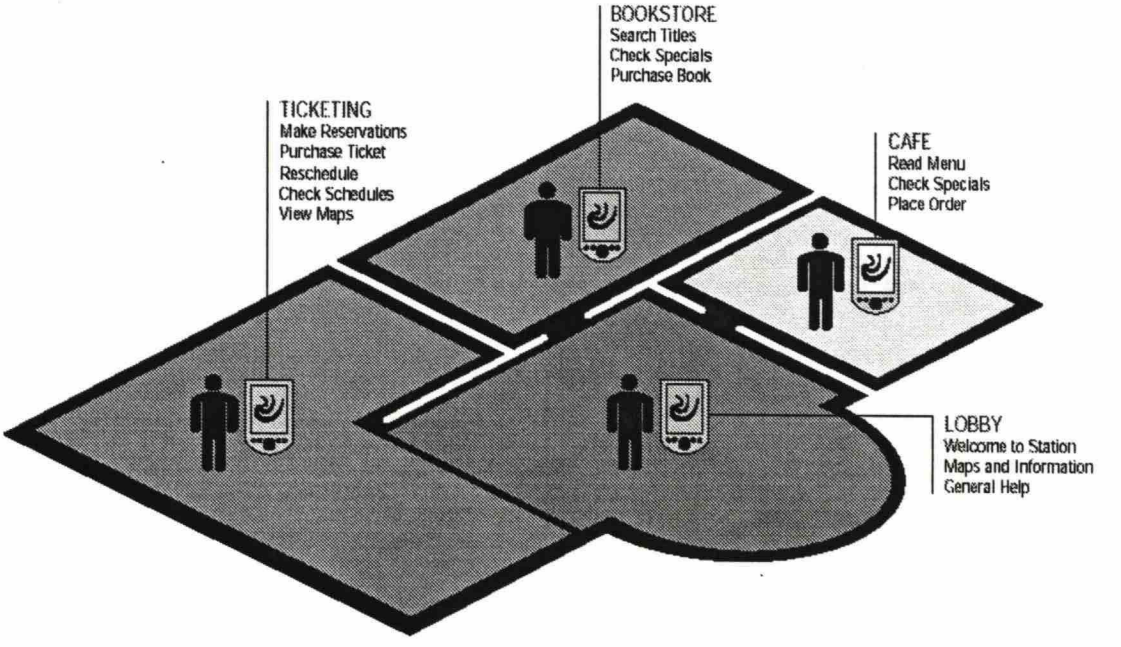


Figure 6. Spaces and services in a train station (Pango, 2001)

3.3.4.2 Actor-Goal list

Table 5. Actor-Goal list for proximity platform

Actor	*Goals*
User	*Discover relevant services in proximity*
Network operator	*Manage spaces & services*
Content provider	*Provide relevant content for end users*
System	*Activate proximity triggered services*

3.3.4.3 Use case stories

Discover services

User arrives to the space. User's mobile terminal shows him a list of available services in his proximity. List is tailored according to User's preferences and location. User selects a service from the list and starts using it.

Manage layout

Wireless network operator deploys the services around the spaces. Each service is mapped to relevant spaces. As content and services are changed, mapping information is updated.

Activate services

As User moves from one space to another System provides a dynamic view of the services and content available in each space.

3.4 Context Analysis

The purpose of the context analysis is to scope the domain that will be further analysed in the domain-modelling phase. The main outputs for this phase are structure and context diagrams. Structure diagram defines the boundaries for the domain and positions it according to other domains. Context diagrams provide more in-depth analysis of the interfaces identified in the structure diagram by describing the actual data flows between the entities. Structure diagram is presented in Section 3.4.1 and context diagrams in Section 3.4.2.

3.4.1 Structure diagram

The purpose of this section is to further scope the domain that was initially defined in the problem statement. It is also important to distinguish the domain from other related items with which it is often confused. In the problem statement, the domain was defined to be part of the more general interactive wireless services domain. More specifically, it was defined to be the domain of context-aware and location-based services. It was also said that a framework should be provided and it should be as independent as possible of the underlying technologies, protocols and platforms. Figure 7 visualizes the scope by showing the position

of context-aware services framework (bolded rectangle) according to the other entities and the concepts of the interactive services domain.

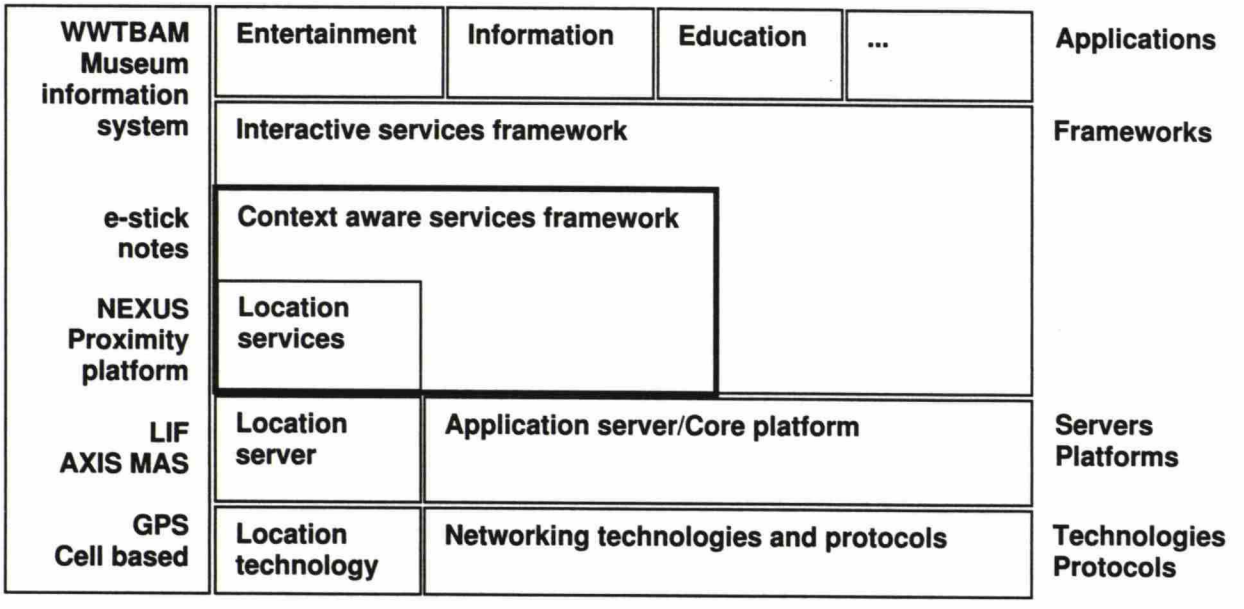


Figure 7. Position of context-aware services framework inside interactive services domain

The right-hand side of the figure shows the different software technology layers that can be identified. Corresponding real world examples for each layer are listed on the left-hand side. It can be seen from the figure that the context-aware services framework is separate from the interactive services framework, which provides more general services. Figure also shows that the framework lies in the middle layer. The lower layers contain the platforms, servers and technologies, which should be transparent for the framework. For these layers, solutions already exist and thus the focus of this paper is on the framework layer. On top of the framework layer is the applications layer. Applications play an important role as a requirement source for the framework. In the next section, we will examine in detail the interfaces and data flows between the identified entities.

3.4.2 Context diagram

The purpose of context diagrams is to show the data flows between the different entities and domains that were identified in the structure diagram. In addition, they should capture the variability of the data flows if there is such.

In the previous chapters the entities that exists in our target domain were identified. General diagram (see Figure 3) showed that there are always a client, a server and a network in between of them. On the server side, there are usually a separate application server and a database. In a location-based environment there is also usually a location server. Section 3.4.1 looked further inside the application server and identified core platform, frameworks and applications. These are the entities to be used in the following analysis. To achieve generalness and to capture the variability of the data flow, analysis will be done separately for two different technologies that are utilized in the target domain. Firstly, the analysis is done for Bluetooth and then for GSM/WAP mobile network. These two have been selected as they are most commonly used for location-based services. After the separate analysis, results are collected into a one generalized diagram at the end.

Bluetooth

When Bluetooth is utilized to implement location-based services (for example see Section 3.3) the following entities can be identified. Network consists of multiple access points, which are connected to the access server (AXIS MAS in the Figure 8) that providing a connection to the application server. Access server provides also the location services and APIs for external systems such as billing and customer care. Clients (PDA) connect to the application server using the nearest access point. Figure 8 shows UML collaboration diagram of the environment describing entities and data flows between them.

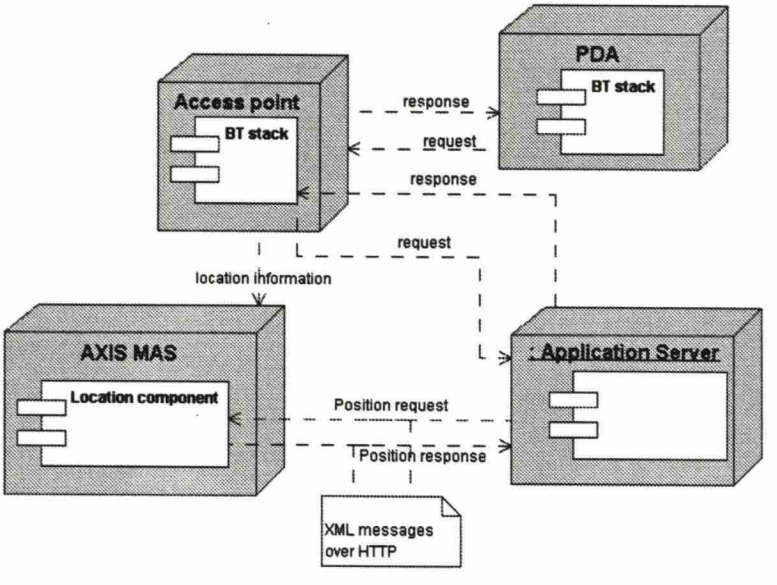


Figure 8. Data flow for Bluetooth network

As the Bluetooth specification work is still going on and the location part is not yet specified a commercial implementation from AXIS (AXIS, 2000) is used as a reference for the diagram.

Typical dataflow of a request for location-based service:

1. Client connects to the access point and sends a request.
2. Request is directed to the application server.
3. Location-based application queries client's location from the access server. In the AXIS case, it is a simple HTTP request containing XML message.
4. Location server knows the access point that client is connected to and returns the location in the response.
5. Application uses the location information and returns a response to the client.

GSM network

In case of the GSM, things look slightly different but the basic entities are the same. Figure 9 shows an UML collaboration diagram for the GSM mobile network. The structure of the GSM network is quite complex and describing it thoroughly is unnecessary here. Thus, only the most relevant entities from the network structure are included in the figure. When GSM is compared with Bluetooth network base station can be seen analogous to the access point of Bluetooth network. One new element in the figure is the WAP gateway that serves as a bridge between the mobile network and the application server, which typically lies in the Internet (WAP Architecture Specification, 2001). Possible realizations of the location server in the GSM network are Gateway Mobile Location Server (GMLC) or Mobile Positioning Center (MPC) (LIF TS 101 Specification, 2002).

Typical dataflow of a request for location-based service:

1. Mobile phone is connected to a base station and sends a request to a WAP service.
2. Mobile network uses WAP gateway to pass the request to the application server.
3. Application server can query client's location from the location server using Mobile Location Protocol (MLP).
4. Location server calculates client's location using typically a cell based technique and returns the location in a response.
5. Application utilizes the location information and returns a response to the client.

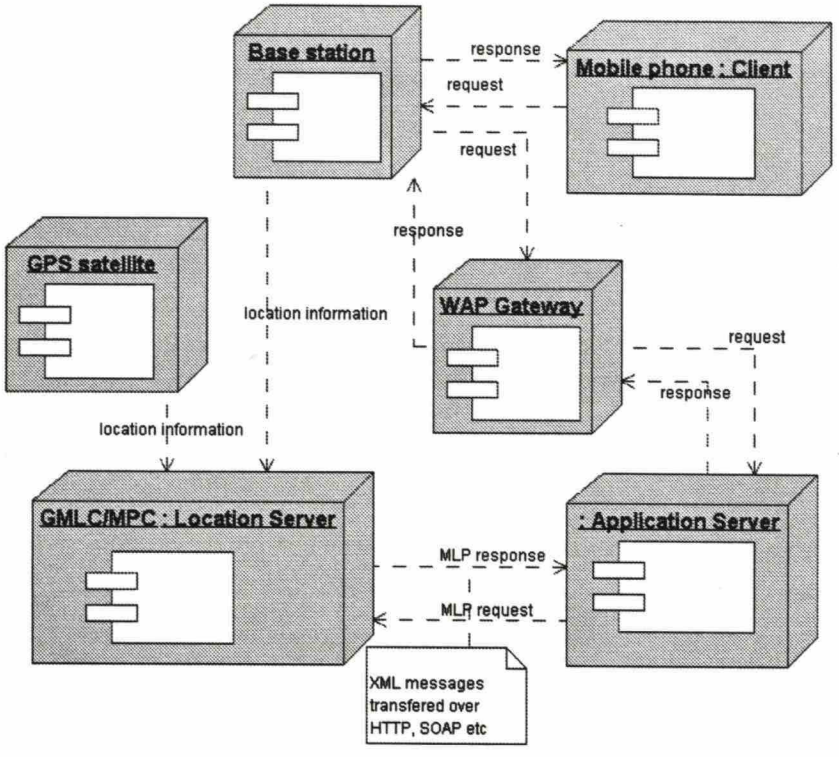


Figure 9. Data flow for GSM/WAP network

Generalized diagram

As we saw from the previous analysis both networks bear lots of similarity in providing location-based services. The generalized model is illustrated in the Figure 10. In the rest of the paper, we will focus looking at this kind of setting where a separate location server exists. Figure also illustrates the data flows inside the application server. The core platform provides basic services such as the connectivity to the outside world and the data access services. When an application needs location or context information, it can use services provided by context-aware services framework. Framework might use the services from the core platform to connect to the location server or the context database.

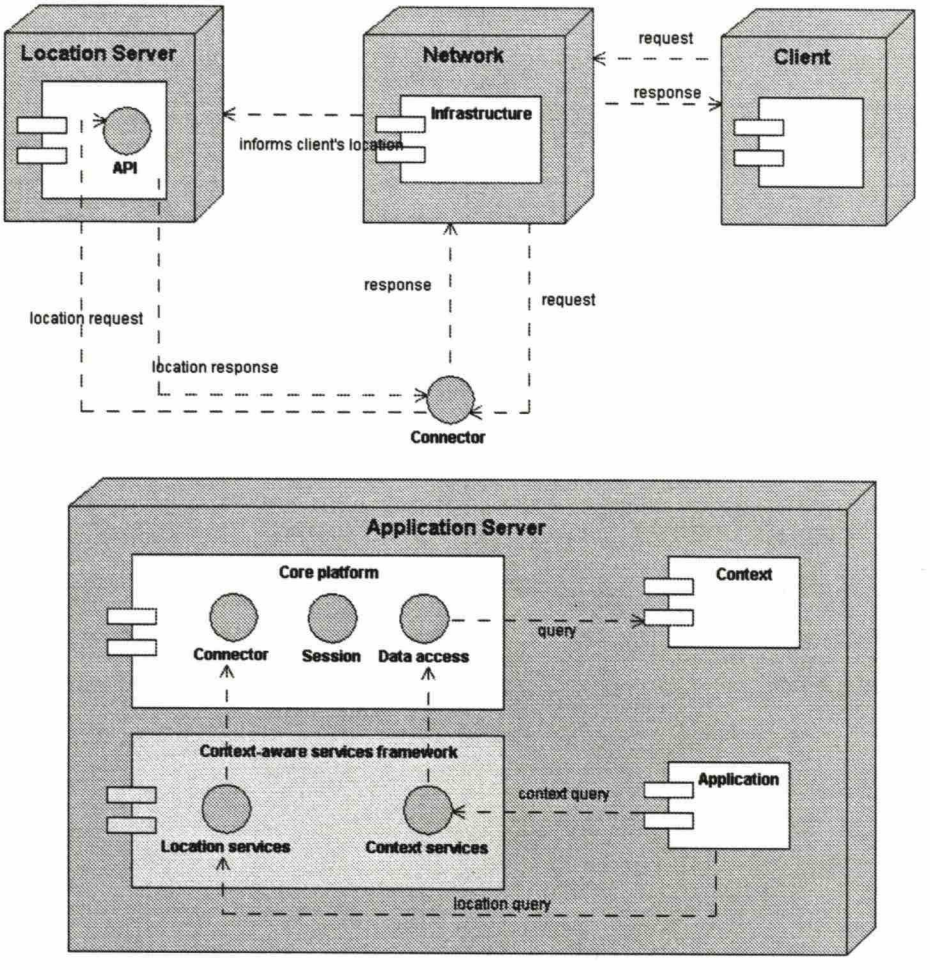


Figure 10. Generalized model of data flow

3.5 Domain modelling

Now that the domain has been scoped, we will next analyse the commonalities and differences of the applications in the specified domain. Domain modelling phase consists of three activities that each provide an analysis from a different aspect: feature and functionality analysis, identifying entities and their relationships and behavioural analysis. Features and functionality are analysed in the Section 3.5.1, which concentrates in context and location - awareness, and Section 3.5.2, where a use case analysis is carried out. Section 3.5.3 identifies entities and their relationships and illustrates them in the domain model. Section 3.5.4 represents a state-chart diagram, a behavioural model of the domain.

3.5.1 Context and location -awareness

In this section, we will analyse the functionality from the context and location -awareness point of view. Typical contexts for applications of this domain will be identified and listed. Analysis was carried out by collecting contexts and features that were mentioned in the applications' documentation. The complete lists of features and contexts can be found from Appendix B. These lists were also used as an input for the use case diagrams that will be presented in the next section.

Analysis revealed that the categorization of contexts suggested in Section 1.2.3 is not enough for the interactive mobile services domain. Some event related contexts were found that do not fit in any of the existing categories. For this reason, a new category "Surroundings context" is proposed to be added. This context is meant to fulfil the needs of the interactive applications that are dependent of the user's social surrounding. Surroundings context captures the user's social surroundings: nearby users, the event user is attending, other activities and their state. For example, when the user is attending a football match the score of the match is part of this context.

Using this new categorisation, typical contexts for this domain can be listed as shown below in the Table 6.

Table 6. Typical contexts for the domain

Computing context
network bandwidth/availability
terminal type
Physical context
- none -
User context
interest
location
access history
profile (age, gender, education, etc)
preferences
Time context
time of day
Surroundings context
social setting
surrounding people
event related context (type of event, match state and result)

3.5.2 Use-Case model

In the Section 3.3 example applications were described using use case stories. In this section, the use cases will be further analysed. Two applications, exhibition tour and stadium entertainment system, were selected as examples and use case diagrams for them can be seen below. Figure 11 shows diagram for exhibition tour and Figure 12 for stadium entertainment system.

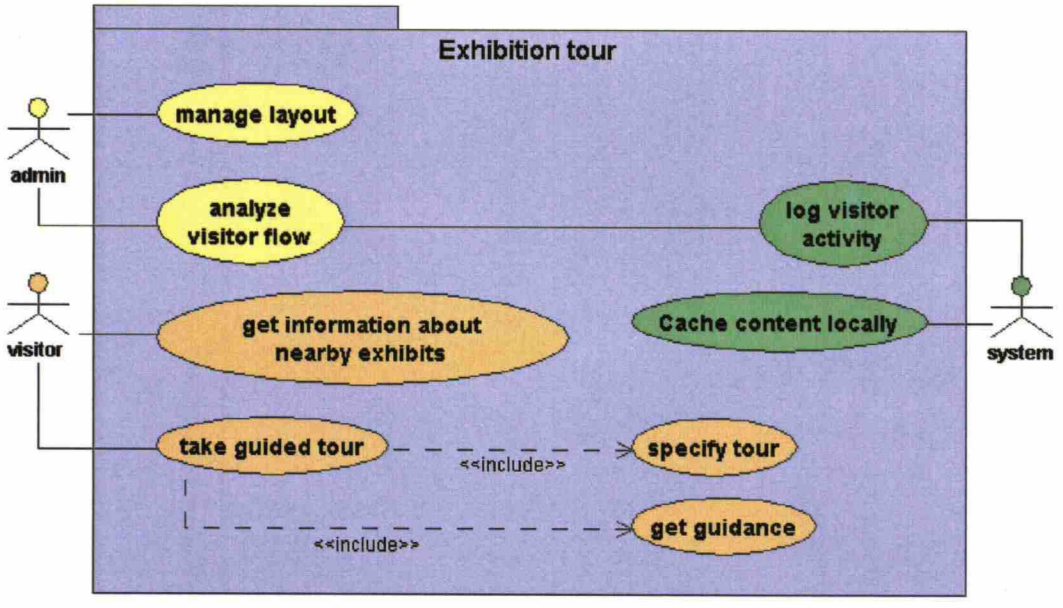


Figure 11. Use case diagram for exhibition tour application

As can be seen from the figures, actors and use cases map to the actor-goal lists that were introduced in Section 3.3. In Figure 11 there is a small difference in guided tour related use cases. These use cases were grouped under one use case "take guided tour". "Specify tour" and "get guidance" use cases, being integral part of the higher level goal, were added under it using the include association.

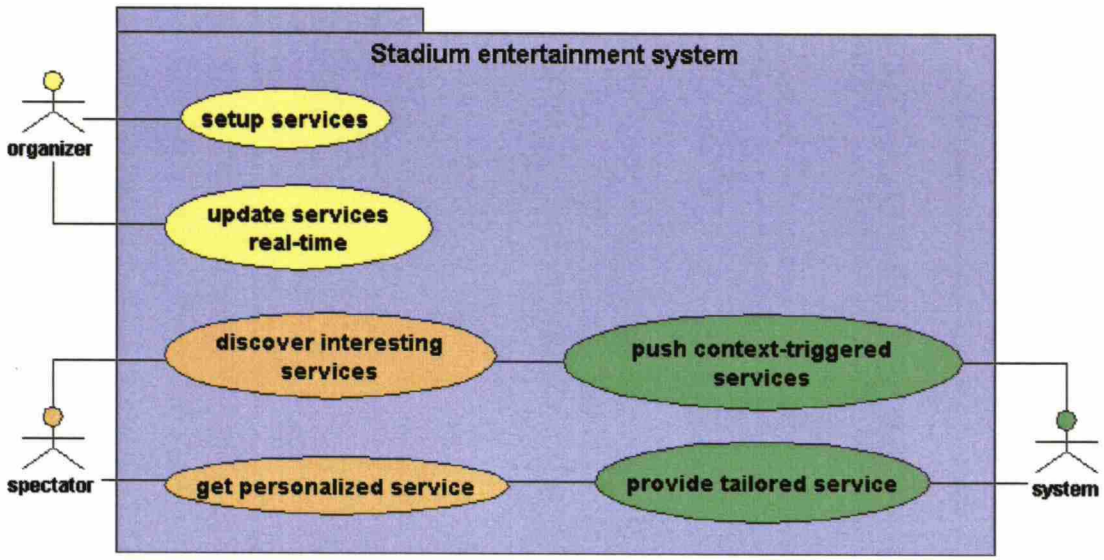


Figure 12. Use case diagram for stadium entertainment system

In Figure 12 "provide tailored service" use case was added to the system actor to emphasize that the system should tailor services according to the spectator's context in order to provide him personalized service.

From these two diagrams a generalized use case diagram can be derived and is shown in Figure 13. It was done by abstracting use cases from the two diagrams to a general form and by removing purely application specific use cases. Generalized use case diagram illustrates the common functionality and features of the applications in this domain. This analysis will be further used in Section 3.6 where some design suggestions are given for a domain-specific software framework. Next, each of the generalized use case will be explained in more detail.

Manage Services

"Manage services" use case consists of two sub use cases: "map services to spaces and contexts" and "update services". Former has the idea that in a location-based application services may be bound to locations (spaces) and more generally in a context-aware application they may be bound to any context. Latter use case involves feature that many times services are updated after the initial setup.

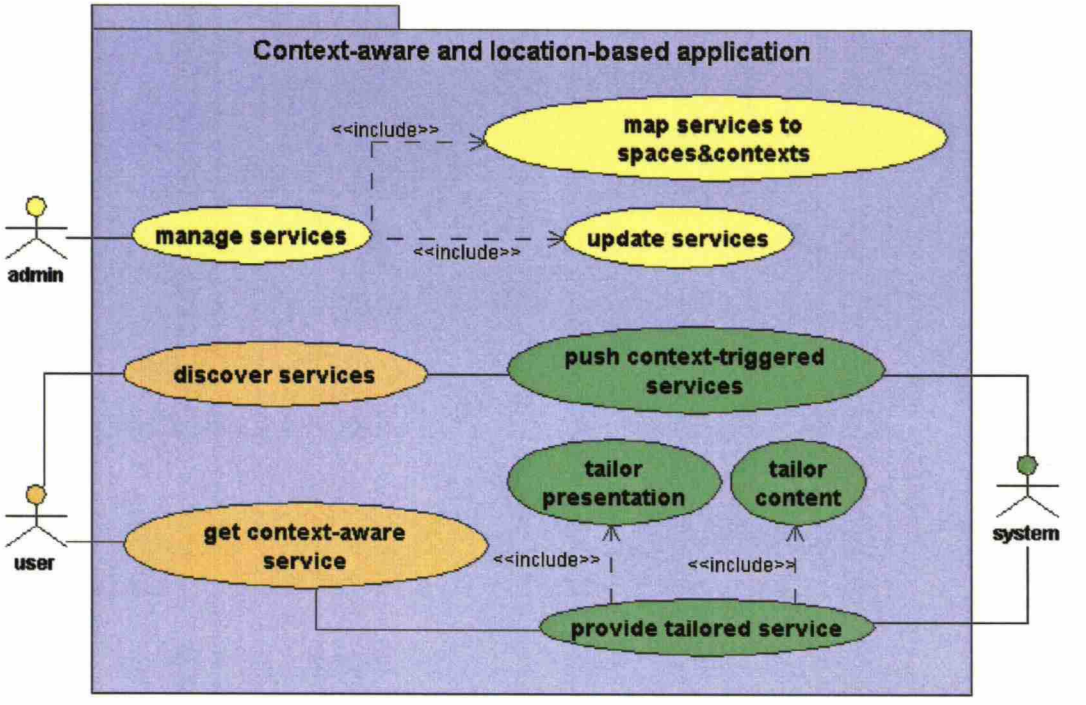


Figure 13. Generalized use case diagram

Discover Services

In order to use services user must first discover them. He may find services from a tailored list or system may automatically trigger them in a certain context, which is explained in the next use case.

Push Context-Triggered Services

In the case that services are mapped to certain contexts system triggers and pushes the service to the user if the specified context is reached. For example, user may enter certain area or certain event may happen and cause the service to activate.

Get Context-Aware Service

The main goal of the user is to get context-aware, personalized service. User expects that the service is tailored according to his preferences and to the current context he is in.

Provide Tailored Service

Providing tailored service includes two parts: tailoring the content and tailoring the presentation. For example, the content may be tailored according to user's profile and the presentation is typically tailored for the terminal that user is using.

3.5.3 Domain Model

Domain model is the most important artifact to create during object-oriented analysis (Larman 2002). It illustrates the meaningful conceptual classes that are used as a source of inspiration for designing software objects. It also provides an easy to understand overview of the problem domain and a vocabulary that can be used in later analysis. Larman suggests the following strategy to identify conceptual classes:

1. Identify noun phrases (use case stories were used as a source)
2. Use conceptual class category lists (see appendix D)

Appendix D lists the conceptual classes that were collected using the above method. Using abstraction and removing less important classes reduced list to 13 classes. These classes constitute the domain model that is illustrated in Figure 14. The entities of the domain model are (in alphabetical order):

Admin: Administrator person specifies and configures services.
Context: Context consists of Place, Event, User, Terminal device and Network for instance. See Section 1.2.3 for more information about context in general.
Event: A thing that happens or takes place. Event can be, for example, a football match, a city tour or an exhibition visit.
Mapping: Mapping maps services to certain contexts and spaces.
Network: In a wireless environment network is typically GSM/WAP network, Bluetooth or WLAN. It hides behind everything that is needed between the end user terminal and the server (GSM Base stations, Bluetooth access points, network infrastructure).
Place: Event takes place in a place, such as a city, an exhibition hall, a train station or a stadium.
SocialSetting: Surrounding social setting (activity, people, etc)
Server: Server hosts the services. In the domain model the concept contains server hardware, platforms and software running in it.

Service: Services are offered to users who use them. See Section 1.2.1 for more information about interactive services.

ServiceDescription: Describes the service.

Space: A space is part of some place where the event is held in. It can be seen as a hot spot where some services can be offered.

TerminalDevice: End user uses terminal device to access the services. Typically in a wireless environment it is a mobile phone, PDA or other handheld device.

User: User is the person who is attending the event and using the offered services.

UsingAService: UsingAService abstracts the concept that at certain point of time somebody is using some service in a certain context with a certain terminal device.

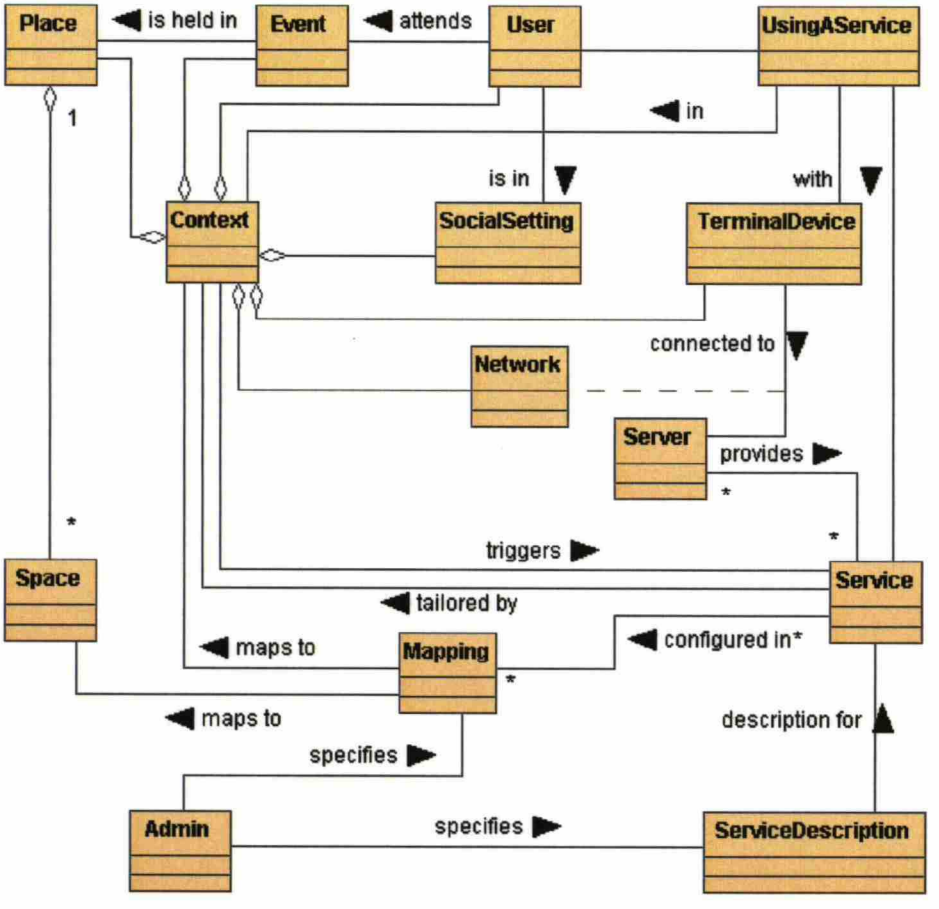


Figure 14. Domain model

After identifying the conceptual classes, associations were added and explained using short verb sentences. Using the words visible in the diagram, the domain model can be explained as follows (black arrows in the diagram show the reading order):

A *User* is in *SocialSetting* and attends an *Event* which is held in a *Place*. He is *UsingAService* in a certain *Context* with a *TerminalDevice*. *TerminalDevice* is connected via *Network* to a *Server* which provides *Services*. *Admin*istrator can specify *Mapping*s that map *Service*s to certain *Context*s and *Space*s. This causes certain *Context*s to trigger *Service*s automatically. *Service*s are described by a *ServiceDescription*.

3.5.4 Behavioural model

Previous three sections analysed the domain from a static point of view. In contrast, this section explores the dynamics and the behaviour of the applications. For this purpose, UML includes state chart diagrams that can be used to illustrate events and states of things (OMG, 2002). Larman (2002) suggests that state chart diagrams should be used to illustrate the external and temporal events rather than the internal events. We will follow this advice in our analysis.

The following analysis is based on the main use cases of the domain: "Discover Services", "Push Context-Triggered Services", "Get Context-Aware Service" and "Provide Tailored Service". Using state chart diagram use cases can be analysed further and events and system's reaction to these events can be explored. Table 7 lists the external events that were identified from the use cases.

Table 7. External events

Event	Description
startup	"startup" of terminal/connection/application
shutdown	"shutdown" of terminal/connection/application
select service	User selects a service
exit service	User exits from a service
request	User (terminal) sends a requests to the service
response	Service responses to the user request
enter hotspot	User enters hotspot area (space)
leave hotspot	User leaves a hotspot
context change	Context changes (clock event, etc)

Using these events different states and transitions between them can be identified. Complete state chart diagram for the domain is shown in Figure 15. Diagram illustrates the two-divided behaviour of the analysed applications. Behaviour can be abstracted into two states: "discover service" and "provide service". First, user must discover a service. Application shows a menu and waits until something causes a service to be activated. "Discover service" state corresponds to use cases "Discover Services" and "Push Context-Triggered Services". After service is activated, application moves to "provide service" state and stays there until the user exits from the service. "Provide service" state corresponds to the use cases "Get Context-Aware Service" and "Provide Tailored Service".

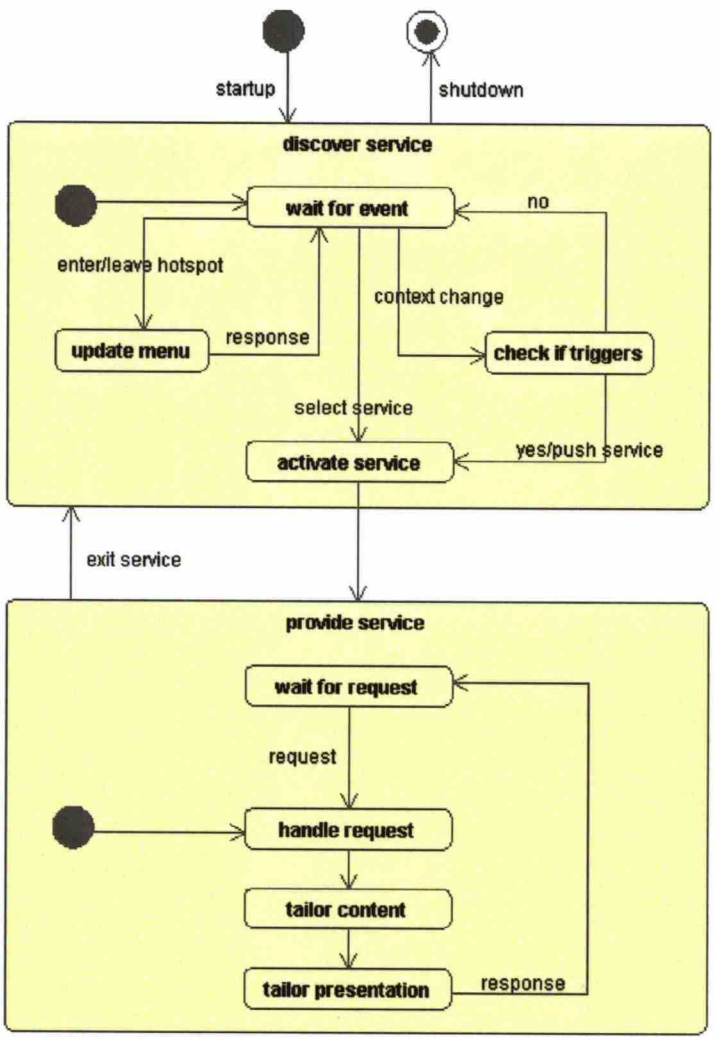


Figure 15. State chart diagram

In the startup, application moves from the initial state to "discover service" state. While inside that state, application stays in "wait for event" state and waits for external events to happen. Events that cause state changes are: "enter hotspot", "leave hotspot", "context change" and "select service". "Enter hotspot" and "leave hotspot" events cause application to move to "update menu" state where service menu is updated according to the new location and a response is sent back to the user. "Context change" event causes application to move to "check if triggers" state where application can check if the change triggers a service. If it does, service is pushed to the user and application moves to "activate service" state. "Select service" is the most typical event and causes simply a state change to "activate service" state. From "activate service" state system moves to "provide service" state.

When system enters "provide service" state, it first handles the initial request and then sends a response to the user and moves to "wait for request" state. Inside "provide service" state application stays in "wait for request" state and waits for a request or "exit service" event that cause application to return to "discover service" state. When a request comes in application first moves to "handle request" state, where application logic happens. Then content and presentation are tailored in "tailor content" and "tailor presentation" states. Finally a response is sent back to user and application moves back to "wait for request" state.

3.6 Architecture modelling

The purpose of the architecture-modelling phase is "to provide a software solution to the problems defined in the domain modelling phase" (Kang et al 1995). The actual implementation is outside the scope of this thesis. Instead, a design reference model for a domain-specific framework will be provided based on the findings from the context analysis and domain modelling phases. These phases identified the common components and behaviour that can possibly be reused in the framework. Having enough commonalities would justify the development of a separate framework. Based on the analysis, four conclusions can be drawn:

1. Context analysis phase → there is synergy in different location-based settings.
2. Context and location -awareness analysis → only certain contexts relevant to the framework.
3. Use case analysis → there is common functionality that can be reused.
4. Behavioural model → there is a need for a common event handling mechanism.

Next, each of these conclusions and their implications for the framework design is analysed further. Figure 16 shows a design reference model that was drawn based on these conclusions. Resources on the left-hand side are hidden behind the framework components that provide interfaces and tools for the services.

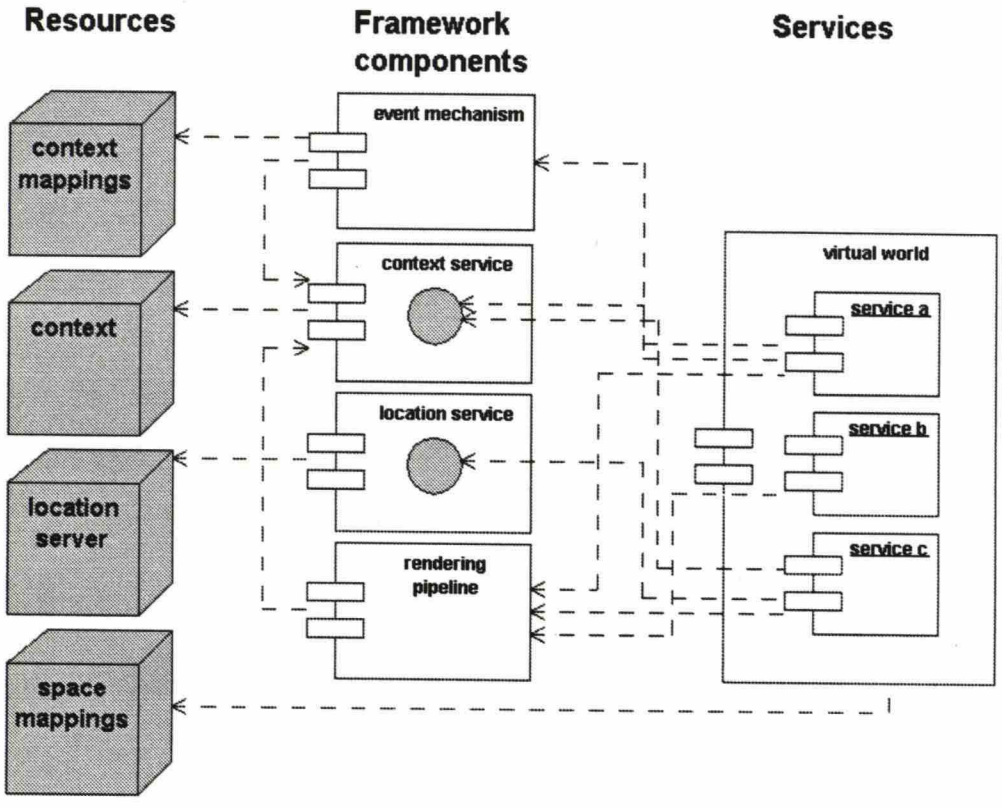


Figure 16. Framework design reference model

Conclusion 1.

As the context diagrams in Section 3.4.2 revealed, there is synergy in how location information is handled in different networks. This suggests that a location service is needed that provides a common interface to location resources (see location service component in Figure 16).

Conclusion 2.

Not all of the contexts listed in Table 6 are relevant for a domain-specific framework. For example, computing contexts such as network bandwidth and terminal type are lower level contexts that are typically taken care of by other frameworks. Similarly, physical contexts such as temperature and lightning are irrelevant for this domain. The most relevant context categories that have to be taken care of by the framework are: User context, Surroundings context and Time context. The dashed ellipse in the Figure 17 illustrates this division of importance.

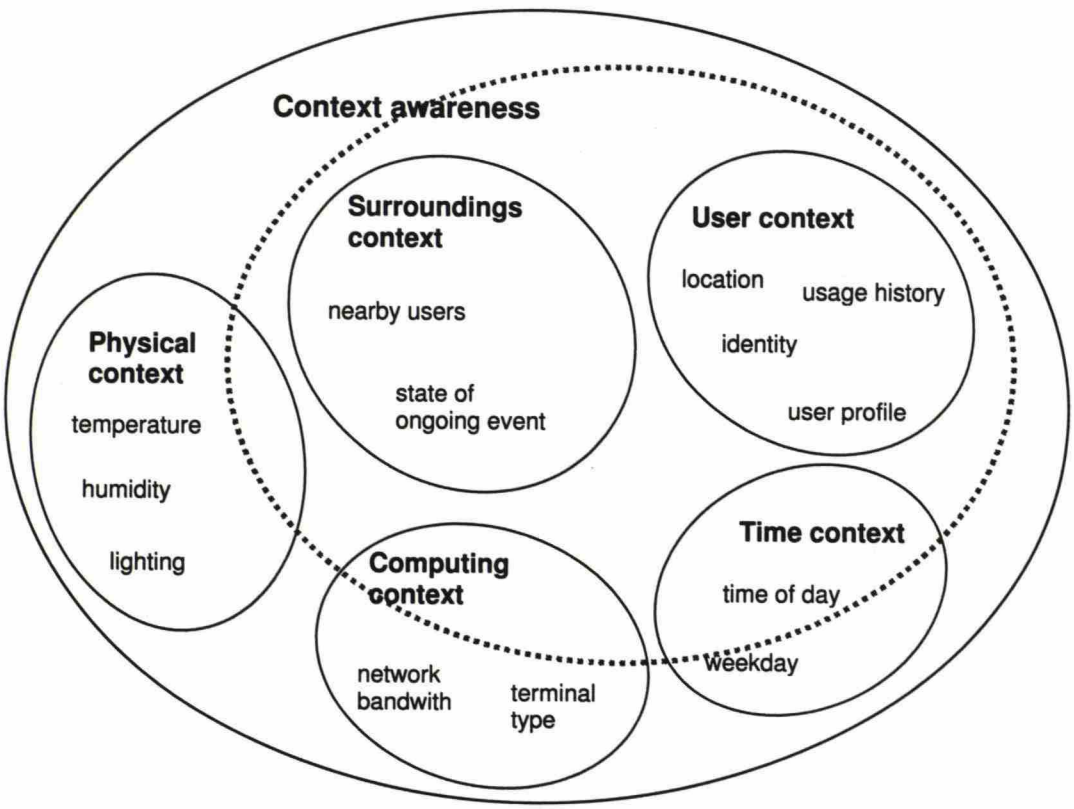


Figure 17. The most relevant contexts for the framework

Conclusion 3.

Use case analysis revealed that common functionality exists in the domain that can possibly be reused in the framework. The following commonalities can be identified from the generalized use case diagram shown in Figure 13:

1. Contextual information (tailoring content & presentation)

   The framework should provide some kind of a context service that can be used to query context information (see context service in Figure 16). Service should provide an interface that hides behind different ways of storing context such as relational database, XML, etc. The framework should also provide a rendering pipeline that can be used to render output according to current context (see rendering pipeline in Figure 16).

2. Context-triggered actions (event → activate service)

   Framework should provide an event mechanism that can be used to trigger services automatically (see event mechanism component in Figure 16). The event handling mechanism will be further designed in conclusion 4.

3. Services mapped to spaces (virtual world)

   The framework should provide a model of a virtual world where services can be mapped to spaces (see virtual world in Figure 16).

Conclusion 4.

The state chart diagram in Figure 15 suggests that the framework should contain an event mechanism, which handles incoming events and reacts to them according to current context. This model is very similar to the ECA-model (Event, Condition, Action) that was originally developed for active databases. In this context these terms can be seen as follows:

- Event is an external event (listed in Table 7)
- Condition means context
- Action means for instance updating menu or triggering a service

Figure 18 presents a collaboration diagram for the component illustrating the various classifier roles and messages between them.

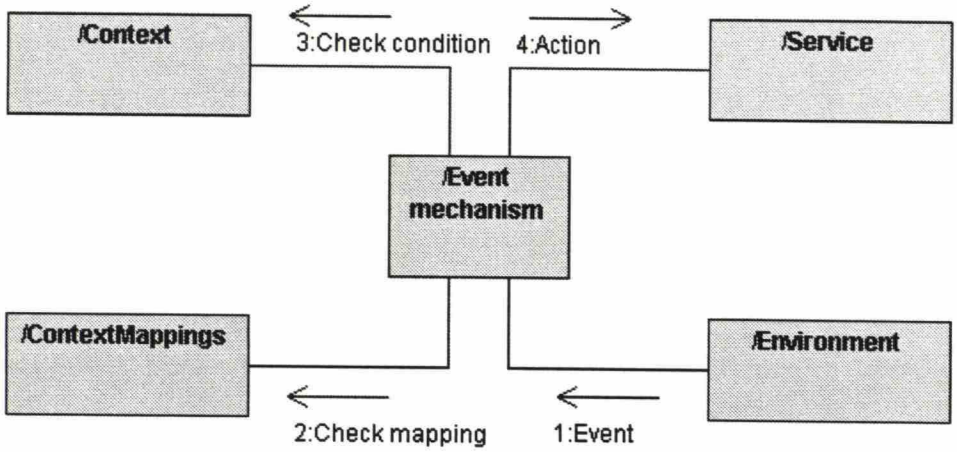


Figure 18. Collaboration diagram of event mechanism component.

4 Evaluation

In this chapter the quality of the solution and the domain model presented in this thesis is evaluated. We try to examine how well the problem defined in the section 1.3 has been solved. The evaluation is based on the criteria stated in the Chapter 2. Before presenting the detailed analysis of each criterion we will next give an overview of the evaluation method.

Criterion 1. stated that the proposed model should be general and applicable to different applications in this domain. To evaluate this, two new applications, that were not used as an input for the analysis are presented and will be next compared with the general model. First the domain model and use case diagram are drawn for both of the applications and then these are compared element by element against the general diagrams. The comparison of domain models should reveal us if application is compatible on a concept level, whereas the use case diagram shows if there is similarity in functionality. State chart diagram is omitted because it is more implementation specific diagram and thus it is not that interesting for the evaluation.

Having two applications enables us to evaluate also the second criterion, completeness of the model. If both of the applications fit the model well, it suggests that completeness has been reached at least on some level. Naturally, evaluating more applications would yield better results for both criterions.

Latter two criterions are more general and thus difficult to evaluate thoroughly. For example, to evaluate understandability in a reliable way would require setting up a test group and evaluating how well these people can use and understand the model. Naturally, this is not possible in this study. Despite these practical problems, we try to evaluate these criterions in some level.

The structure for this chapter is as follows. In Section 4.1, the applications that were used in the evaluation are introduced. After that, the evaluation against each criterion is carried out in Section 4.2. Finally, Section 4.3 summarizes the evaluation and points out some remarks regarding architecture modelling.

4.1 Evaluated applications

The first application that is used in the evaluation is the Conference Assistant presented in the paper "The Conference Assistant: Combining Context-Awareness with Wearable

Computing" (Dey et al, 2000). Conference assistant is a mobile, context-aware application that assists conference attendees. It helps attendee to organize his schedule, find interesting presentations for him and to make notes, for instance. The second application is a "FieldNote: a Handheld Information System for the Field" (Ryan et Pascoe, 1999). It is a data collection tool for fieldworkers such as environmental scientists and archaeologists. System consists of a handheld device that is used for data collection on site, a server that stores the data and a client application that can be used to analyse the data afterwards.

Due to lack of existing applications in this domain we had to choose these two despite neither of them is exactly from the domain. Conference assistant does not fit the client-server based model that was described in Section 3.2 and FieldNote application is more a tool than a service. However, these applications are close enough so that they can be used to analyse the model from context-awareness point of view. It is also interesting to see how well they fit the model nevertheless being somehow outside the domain.

As mentioned earlier, use case diagram and domain model were drawn for both of the applications. Figure 19 shows a use case analysis of the Conference Assistant application and Figure 20 presents a domain model for it. Figure 21 and Figure 22 present the corresponding diagrams for the FieldNote application. Next, each of the criterions are analysed using the diagrams.

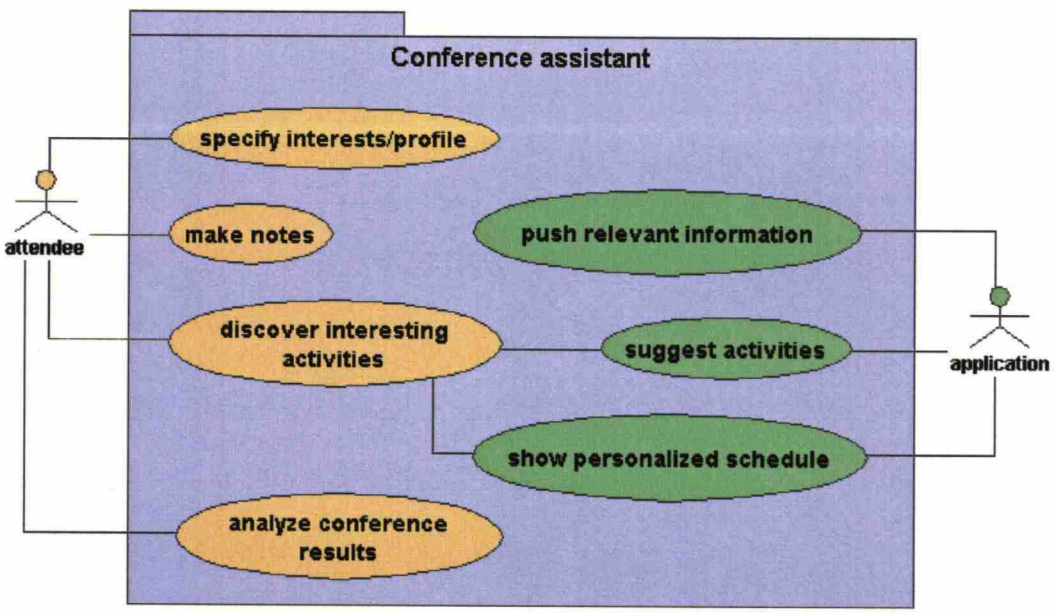


Figure 19. Use case diagram for Conference Assistant application

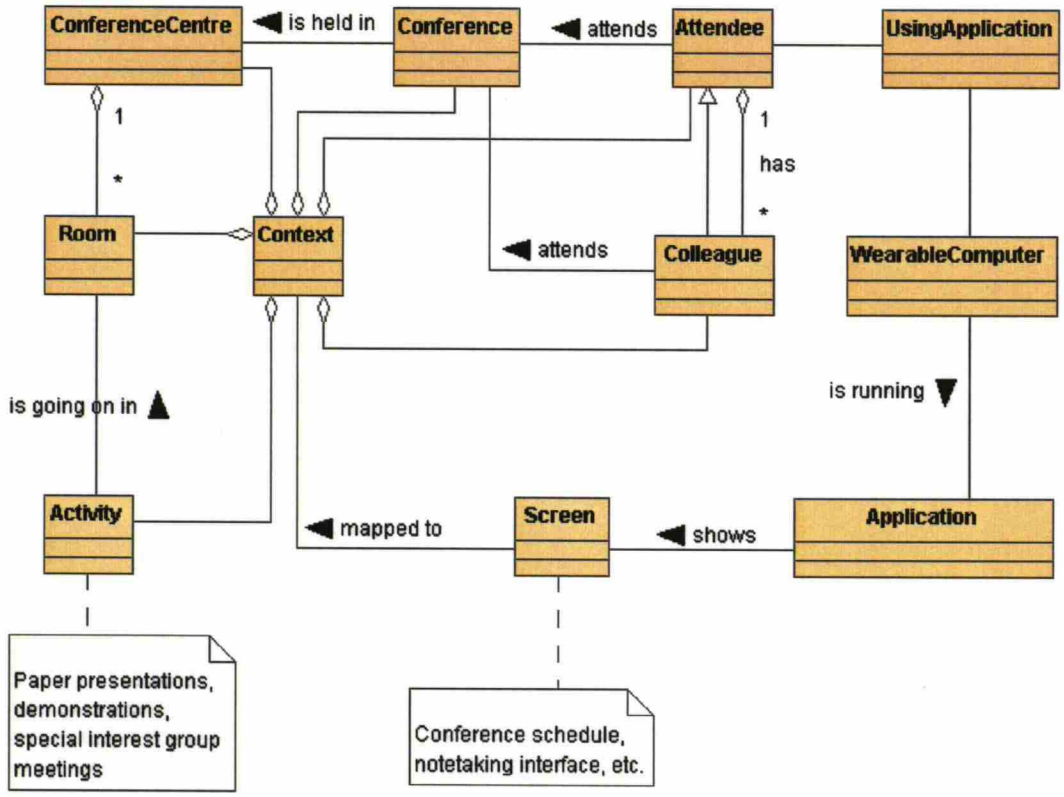


Figure 20. Domain model of Conference domain

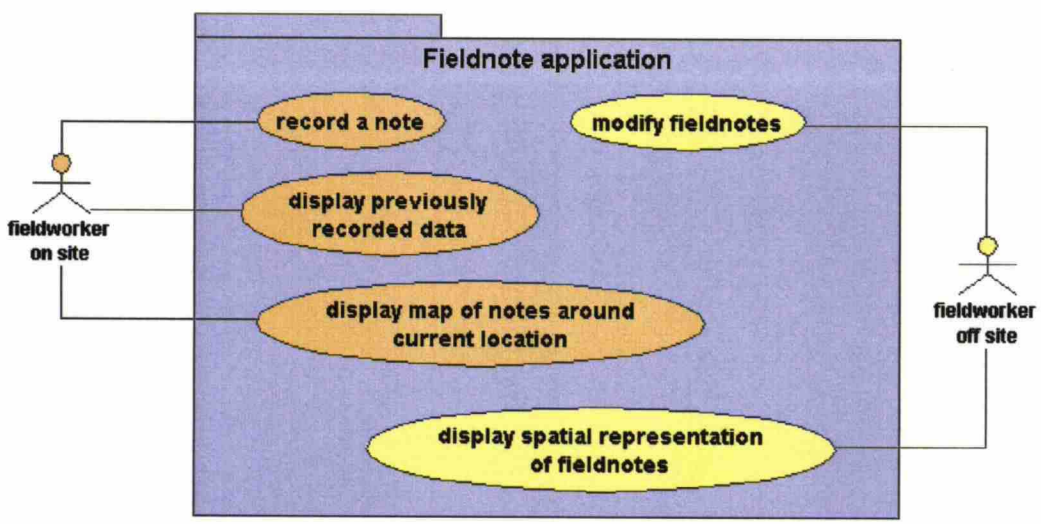


Figure 21. Use case diagram for FieldNote application

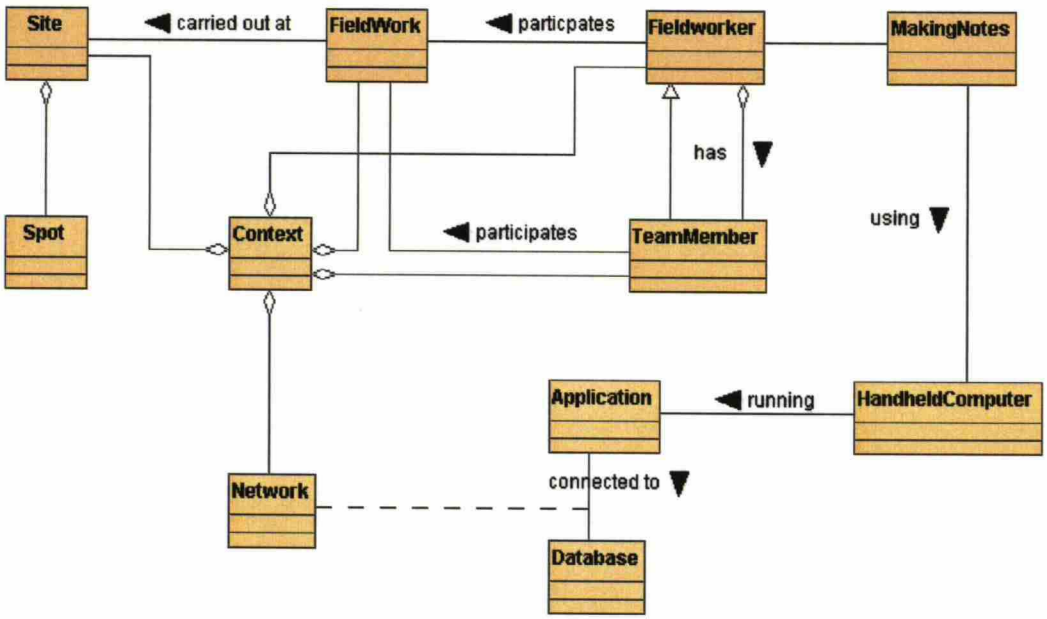


Figure 22. Domain model of FieldNote application

4.2 Evaluation against each criterion

Criterion 1. Generality and Applicability

The evaluation of generality and applicability of the model is done by analysing how well each element of the diagrams matches with the general model. The results for the use case diagram can be seen in Table 8 and respectively for the domain model in Table 9. For each element of the diagram, there is a corresponding row in the table. Rows are also coloured indicating the level of the match as follows:

Green: There is a match between the application and the general model.

Yellow: Issue has been addressed but there is no exact element match in the general model.

Red: No matching concept exists in the general model.

Table 8. Use case diagram match tables.

Conference Assistant	
Element	*Match*
discover interesting activities	*In the general model there is a discover services use case that illustrates the same idea that user wants easily discover interesting and relevant things for him.*
make notes	*There is no matching use case for this type of context-aware recording of things.*
specify interests/profile	*Specify guided tour use case for Exhibition tour matches with this.*
analyze conference results	*Analyze visitor flow use case for Exhibition tour matches with this. Activity(history) of user(s) is analyzed.*
push relevant information	*General model contains push context-triggered services use case that addresses the same issue*
suggest activities	
show personalized schedule	*General model has provide tailored service (tailor content) use case which can be seen as a generalization of this.*

Fieldnote Application	
Element	*Match*
record a notes	*There is no matching use case for this type of context-aware recording and modifying of things.*
modify fieldnotes	
display spatial representation of fieldnotes	*Analyze visitor flow use case for Exhibition tour matches with this. Activity(history) of user(s) is analyzed.*
display previously recorded data	*General model has provide tailored service use case which can be seen as a generalization of this (Tailor content using location and history contexts).*
display map of notes around current location	

Table 9. Domain model match tables.

Conference Assistant	
Element	*Match*
ConferenceCenter	*Place is a generalization of this.*
Conference	*Event is a generalization of this.*
Attendee	*User is a generalization of this.*
UsingApplication	*UsingAService is a generalization of this.*
Room	*Space is a generalization of this.*
Context	*Maps exactly with the Context element in the general model.*
Colleague	*Part of the SocialSetting element in the general model.*
WearableComputer	*One kind of a TerminalDevice*
Activity	*Part of the SocialSetting element in the general model.*
Screen	*Do not match with the client/server model of the general diagram.*
Application	

Fieldnote Application	
Element	*Match*
Site	*Place is a generalization of this.*
Fieldwork	*Event is a generalization of this.*
Fieldworker	*User is a generalization of this.*
MakingNotes	*UsingAService is a generalization of this.*
Spot	*Space is a generalization of this.*
Context	*Maps exactly with the Context element in the general model.*
TeamMember	*Part of the SocialSetting element in the general model.*
HandheldComputer	*One kind of a TerminalDevice*
Network	*Maps exactly with the Network element in the general model.*
Application	*Do not match with the server side services*
Database	*Part of the server in the general diagram*

We can see above that green is dominant colour in the tables. This suggests that the general model applies well for these applications. It is normal that each application has some application specific variation and the match is not perfect. This explains most of the red and yellow rows in the tables. Next, these rows are analysed in more detail.

Use case diagram match tables:

make notes, record a note

This can be seen as a common feature of saving information together with the current context. It is completely missing from the general model. Reason for this is that the model has been developed for services that typically provide information rather than collect it. However, it would be a good idea to somehow address this kind of behaviour.

modify field notes, display spatial representation of field notes, analyse conference results

These are administrative features that have been omitted from the model. Similar things came up also with other applications but were considered too specific features. This can be seen as normal application variation.

specify interests/profile,

User profile is seen as very important context in the model. Specify own tour/guide in tourist guide application is very similar to this.

Domain model match tables:

Colleague, Team member, Activity

These can be seen as being part of the SocialSetting element in the general model. Applications that were used in the original analysis are single user applications which explains why the concept of a colleague, a team member or similar did not come up separately. Based on these new findings it would be justified to model it as its own element.

Screen, Application, Database

Because the general model was developed for server based services model, it is normal that client side applications do not fit the model exactly.

As a summary, we can say that domain models match really well with each other. Only 3 elements out of 22 do not match at all and 15 elements have a corresponding concept in the general model. We can also claim that the proposed model is general and applicable to different applications. Looking at the elements in the domain model we can also see that they are very general terms and there are no specific elements for certain technologies. This is also in accordance with the criteria of generality.

Criterion 2. Completeness

Because the model covered both of the evaluated applications rather well we can claim that it fulfils the criterion of completeness as well. There were only few elements that were not matched. The model also fulfils the other aspect of completeness in a sense that it covers different aspects of an application. Context diagram shows messaging with other entities, use case diagram covers the common functionality, domain model presents the different concepts and their relationships and state chart diagram illustrates the behaviour of the applications.

Criterion 3. Understandability

Most of the words used in the diagrams are such that can be heard in everyday language. This should make it easy to understand the model and find the matching concepts for a specific application. However, to understand the diagrams user must be somehow familiar with object-oriented design and UML modelling. Apart from that, we can say that model is easily understandable.

Criterion 4. Consistency

The whole work is based on existing standards and the work that has been done so far in the research world. UML were used as a modelling language whenever it was possible. This guarantees the consistency of this work.

4.3 Summary of evaluation

In the problem statement in Section 1.3 two objectives were defined: to analyse the problem domain and to produce a domain model, and to propose some design guidelines for a domain-specific software framework. Based on the evaluation in the previous section we can say that the first objective has been reached and the quality of the solution is according to the defined criteria. The second objective was addressed in the Section 3.6 Architecture modeling. That section explains why the development of a domain-specific framework is justified. Based on the findings from the analysis, the main components of the framework were identified and general guidelines were given regarding their implementation. This fulfils the second objective of this thesis.

5 Conclusions

The main objective of this thesis was to develop a domain model for the location-based and context-aware mobile services domain. The other objective was to use the produced domain model and suggest some design guidelines for a domain-specific software framework.

As a method for the analysis, the Feature Oriented Domain Analysis (FODA) combined with the unified modelling language (UML) was selected. Following the method, four example applications were selected and analysed together with the existing standards and earlier research work. As a result, a general model of the domain was produced consisting of context analysis; use case analysis, domain model and behavioural model. Finally, based on the analysis, some architecture modelling was done providing design guidelines and suggestions for developing a domain-specific software framework.

The main achievement of this thesis is that it presents a complete and easily understandable model of the domain. Many papers have studied context-aware applications but none of them has presented a structured model based on the industry's modelling standard UML. This paper also has a new and interesting viewpoint on context-awareness. The application domain studied has one distinguishable characteristic: the importance of surrounding social activity. This hopefully provides some new ideas for the context-awareness research.

This thesis also serves as a good example of an analytical domain analysis. It shows how the UML can be used for domain modelling in a practical way. It also shows how useful the use cases are. Written use cases are a great tool and source of inspiration for later analysis. It turned out that many elements originally found in the use case analysis made their way into the final model. Use case analysis was also found to be a very straightforward and simple method that still produces a rich model. Some other advantages are that it is practical and not too time consuming. Both are properties that are well appreciated in the software industry.

The lack of existing applications in the domain restricted the analysis and thus limited the presented solution. Even though the model does not cover everything, it still identifies the most typical properties of the applications and naturally new properties can be added later if needed. Another limitation of the solution is that it does not say much about the implementation. Some design guidelines and suggestions were given but analysis of the results in greater depth is required before a domain-specific software framework can be implemented. This is one of the tasks left for future work. It includes work such as

identifying common abstractions to be implemented in the framework and deciding how variations should be handled.

In the field of context-awareness, there are still quite many open issues to be resolved in the future. Some of these are: How should the context information be stored? What would be the standard format to present context? How could services share the context information? How are changes in the context sensed? The last question is particularly interesting for this domain. For example, how do applications know if there is a change in the surrounding social activity?

Despite there are still some unsolved issues, the activity in the research world and the market in general proves that context-awareness is coming and will be one of the key factors for the success of future mobile services.

References

Abowd, G.D.; Atkeson, C.G.; Hong, J.; Long, S.; Kooper, R.; Pinkerton, M. 1997. Cyberguide: A Mobile Context-Aware Tour Guide. ACM Wireless Networks, 3. pp. 421—433.

Axis Communications. 2000. The Mobile Internet and Wireless Networking: Opportunities, Challenges and Solutions. Referenced 22 March 2002. http://www.axis.com/documentation/whitepaper/wireless/mobile_access.pdf.

Baum, L.; Becker, M.; Geyer, L.; Gilbert, A.; Molter, G.; Tamara, V. 2000. Supporting Component-Based Software Development Using Domain Knowledge. Proceedings of the SCI 2000 Conference. Orlando, Florida, USA, July.

Cockburn, A. 2001. Writing Effective Use Cases. 1st edition. Addison-Wesley. Reading, MA. ISBN 0-201-70225-8.

Chen, G.; Kotz, D. 2000. A Survey of Context-Aware Mobile Computing Research. Technical Report TR2000-381. Dept. of Computer Science, Dartmouth College.

Cheverst, K.; Davies, N.; Mitchell, K. et al. 2000. Developing a Context-aware Electronic Tourist Guide: Some Issues and Experiences. In Proceedings of CHI'00, Netherlands, April, ACM Press. pp 17-24.

Dey, A. K.; Abowd, G. D. 1999. Towards a Better Understanding of context and context-awareness. Technical Report GIT-GVU-99-22. Georgia Institute of Technology, College of Computing.

Dey A., K.; Futakawa M.; Salber D.; Abowd G., D. 1999. The Conference Assistant: Combining Context-Awareness with Wearable Computing. Proceedings of the 3rd International Symposium on Wearable Computers (ISWC '99). San Francisco, CA, October 1999. IEEE Computer Society Press. Pages 21—28.

Dey, A.K.; Salber, D.; Abowd, G.D. 2001. A Conceptual Framework and a Toolkit for Supporting the Rapid Prototyping of Context-Aware Applications. Human-Computer Interaction (HCI) Journal. Vol. 16(2-4). pp. 97—166.

Froehlich, G.; Hoover, J.; Liu, L.; Sorenson, P. 1998. Designing object-oriented frameworks. CRC Handbook of Object Technology. CRC Press.

Kang, K.; Cohen, S.; Hess, J.; Peterson, W.; Peterson, S. 1990. Feature-Oriented Domain Analysis (foda) Feasibility Study. Technical Report CMU/SEI-90-TR-21. Software Engineering Institute, Carnegie Mellon University. 148 p.

Kishimoto M.; Takada Y.; Oiso H.; Komoda N.; Yamasaki T.; Masanori T. 2001. The development of information service system using Bluetooth in an exhibition hall. In Proc. of 4th International Conference on Electronic Commerce Research (ICECR-4). Dallas, Texas, USA, August 8-11, 2001. Vol.1. pp.178—183.

Larman, G. 2002. Applying UML and Patterns: an introduction to object-oriented analysis and design and the Unified Process. 2nd Edition. Upper Saddle River, NJ 07458. Prentice Hall PTR. 627 pages.

Manes A., T. 2001. Sun Microsystems. Enabling Open, Interoperable, and Smart Web Services. The Need for Shared Context. Last modified 12 March 2001. Referenced 22 January 2002. http://www.w3.org/2001/03/WSWS-popa/paper29

LIF TS 101 Specification. Version 3.0.0. Location Inter-operability Forum (LIF). Last modified 6 June 2002. Referenced 22 January 2002. http://www.locationforum.org/publicdownload/LIF-TS-101-v3.0.0.zip

Microsoft .NET Services. Microsoft. Last Modified 19 June 2002. Referenced 22 March 2002. http://www.microsoft.com/netservices/

OMG Unified Modeling Language Specification. Version 1.4. September 2001. Last modified 21 January 2002. Referenced 20 March 2002. http://www.omg.org/technology/documents/formal/uml.htm

Park, A. S.; Lipperts, S.; Wilhelm, M. 2002. Location Based Services for Context Awareness – Moving from GSM to UMTS. Proceedings of the International Conference on Advances in Infrastructure for Electronic Business, Science, and Education on the Internet. SSRGG2002W. L'Aquila, Italy. February 21-27, 2002.

Proximity Platform. Pango Networks. Referenced 11 January 2002.. http://www.pangonetworks.com

Ryan, N.; Pascoe, J. 1999. FieldNote: a Handheld Information System for the Field.

Schilit, B.; Adams, N.; Want, R. 1994. Context-aware computing applications. Proceedings of IEEE Workshop on Mobile Computing Systems and Applications. Santa Cruz, California. December 1994. IEEE Computer Society Press. pp 85—90.

Software Technology Review. Carnegie Mellon University. Last Modified 11 June 2001. Referenced 22 January 2002. http://www.sei.cmu.edu/str/

WAP Architecture Specification. Version 12-July-2001. WAP-210-WAPArch-20010712. Referenced 20 March 2002. http://www.wapforum.org/what/technical.htm

Whatis?com. Referenced 23 August 2002. http://www.whatis.com

Appendix A, Standards and abbreviations

Abbreviation	*Explanation*
API	*Application Programming Interface*
FODA	*Feature-Oriented Domain Analysis*
GMLC	*Gateway Mobile Location Server*
GPS	*Global Positioning System*
HTTP	*HyperText Transfer Protocol*
LAN	*Local Area Network*
LBS	*Location-Based Services*
LIF	*Location Inter-operability Forum*
MPC	*Mobile Positioning Center*
PDA	*Personal Digital Assistant*
SMS	*Short Messaging Service*
UML	*Unified Modelling Language*
UMTS	*Universal Mobile Telecommunications System*
UP	*Unified Process*
WAP	*Wireless Application Protocol*
WLAN	*Wireless LAN*
W3C	*World Wide Web Consortium*
XML	*eXtended Markup Language*

Appendix B, Feature lists

Exhibition Tour:

Type	*Description*
Feature	*automatic explanation of a nearby exhibit*
Feature	*support for both voice and data information*
Feature	*specification of personal tours*
Feature	*recommended tours*
Feature	*tour guidance*
Feature	*analysis of visitors' flow from room to room*
Feature	*exhibition (layout) specification/modification*
Feature	*provide additional information using hyperlinks*
Context	*visitor's interest*
Context	*terminal type*
Context	*visitor's location*
Context	*access history*
Context	*user profile (age, gender, education, etc)*
Context	*network bandwith/availability*

Stadium Entertainment System:

Type	*Description*
Feature	*support for different type of services (information, entertainment, surveys)*
Feature	*triggered/scheduled actions/services*
Feature	*different services in different contexts and locations*
Feature	*content/questions in the service(trivia) change according to context*
Feature	*automatic service discovery*
Context	*location*
Context	*user profile (age, gender, favourite team)*
Context	*time of day*
Context	*event related context (type of event, match state and result)*
Context	*network bandwith(streaming video)*

Tourist Information System:

Type	*Description*
Feature	*map component (physical surroundings, map)*
Feature	*information component (information about sights)*
Feature	*location component (user's location)*
Feature	*communication component (tourist wants to ask something from TIC, broadcast something to group of tourists)*
Feature	*active component that can react to events*
Feature	*suggests events happening in near future (scheduled events)*
Feature	*capture user's thought, reactions and feedback*
Feature	*Automatic authoring of a travel diary*
Feature	*Utilization of existing content (for example from the web)*
Feature	*information tailored to both personal and environmental context*
Feature	*flexibility to explore and learn about in their own way*
Feature	*control the pace of interaction, interrupts*
Feature	*presentation of information tailored to the context*
Feature	*dynamic information (notifications)*
Feature	*hypertext information*
Feature	*cache locally part of the information*
Feature	*information retrieval*
Feature	*navigation in the city*
Feature	*creating and then following the tour(take into account closing and opening times, distance between attractions, best time to visit)*
Feature	*view overview map or map of local area*
Feature	*tell visitor about the area*
Feature	*answer questions on something visitor can see*
Feature	*tell visitor the weather forecast*
Feature	*tell visitor about news and events in the city*
Feature	*contents page of all information*
Context	*how person reacted to other sights*
Context	*interest*
Context	*visitor's location*
Context	*history (of visited sights)*
Context	*refreshment preferences*
Context	*where others are*
Context	*time of the day*
Context	*weather*
Context	*opening times of attractions*

Proximity Platform:

Type	*Description*
Feature	*mapping spaces to a physical area*
Feature	*personalized, localized, timely content*
Feature	*Proximity triggered services (enters, exits, moves within, time passes)*
Feature	*binding services to spaces*
Context	*User preferences*
Context	*User's location*
Context	*User profile*
Context	*Time of day*

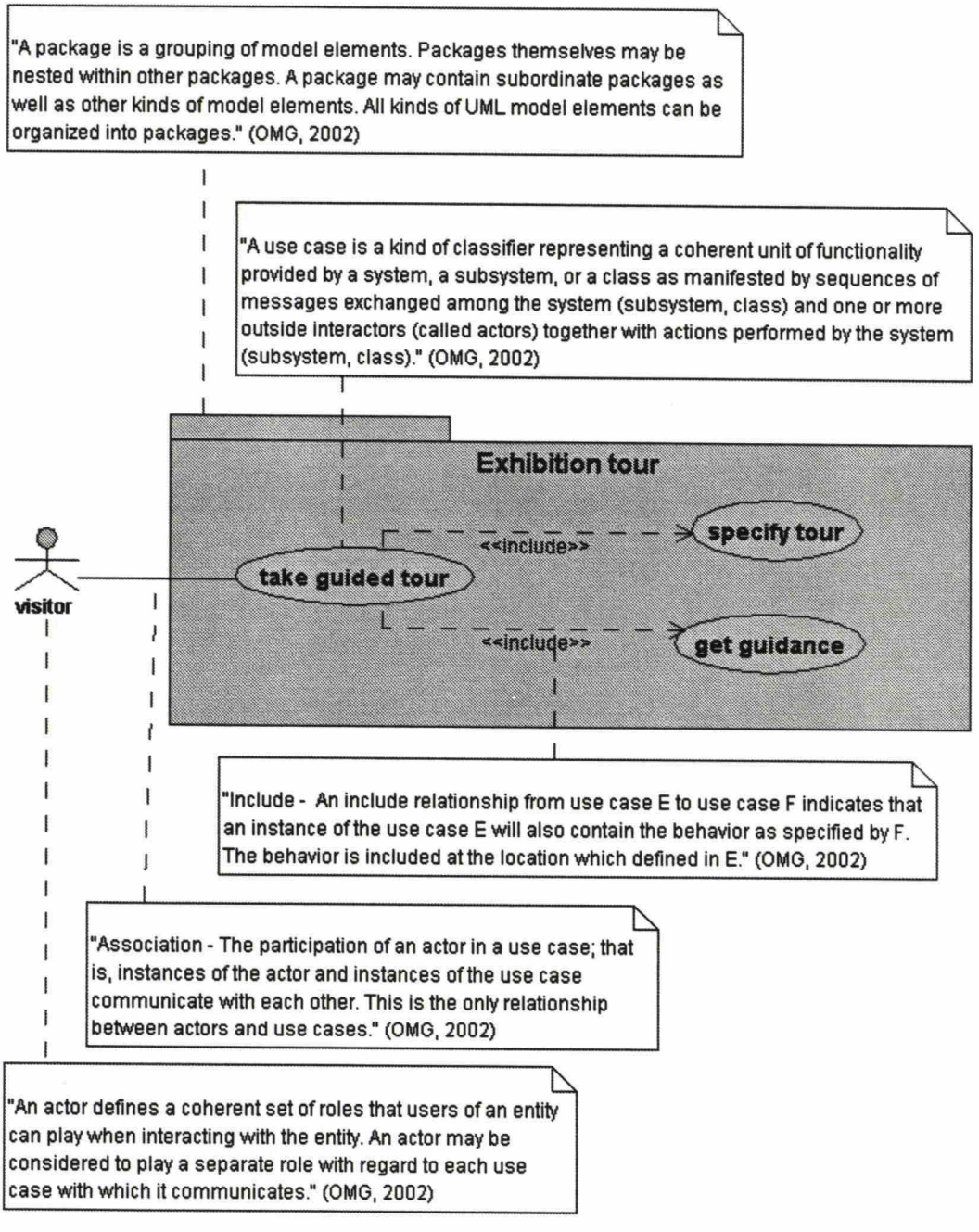
Appendix C, Conceptual classes in categories

Conceptual Class Category	*Examples*
Physical or tangible objects	*Terminal, Screen, Exhibit, Attraction, Network, System, Service, Platform*
Specifications, designs, or descriptions of things	*Service type (Promotion, Advertisement, Competition, Information, Entertainment), Schedule*
Places	*Hall, Stadium, Museum, Exhibition, City, Station, Store, Destination, Area, Hot-spot*
Transactions	*Request, Response*
Roles of people	*Tourist, Visitor, Manager, Administrator, Spectator, Event Organizer*
Containers of other things	*List, Map, Layout, Content(Text, Audio, Video clip), Menu*
Abstract noun concepts	*Direction, Route, Space, Context(Location, Time, Profile, Interest), Environment*
Organizations	*TIC*
Events	*Exhibition, Match, Sport Event, Tour, Visit, Activity*
Processes	*Booking, Guiding, Presentation*
Books	*Guide book*

Appendix D, UML syntax

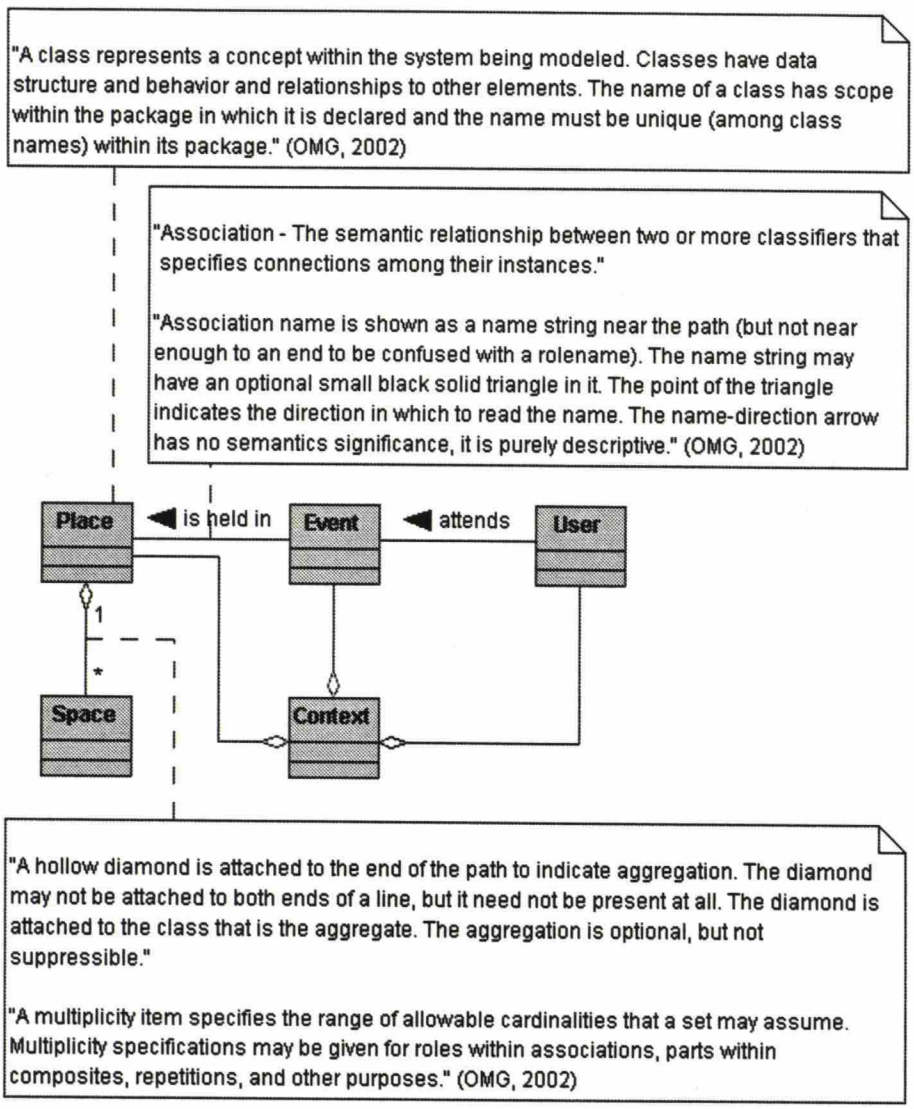
Use case diagram:

"A use case diagram is a graph of actors, a set of use cases, possibly some interfaces, and the relationships between these elements. The relationships are associations between the actors and the use cases, generalizations between the actors, and generalizations, extends, and includes among the use cases. The use cases may optionally be enclosed by a rectangle that represents the boundary of the containing system or classifier." (OMG, 2002)



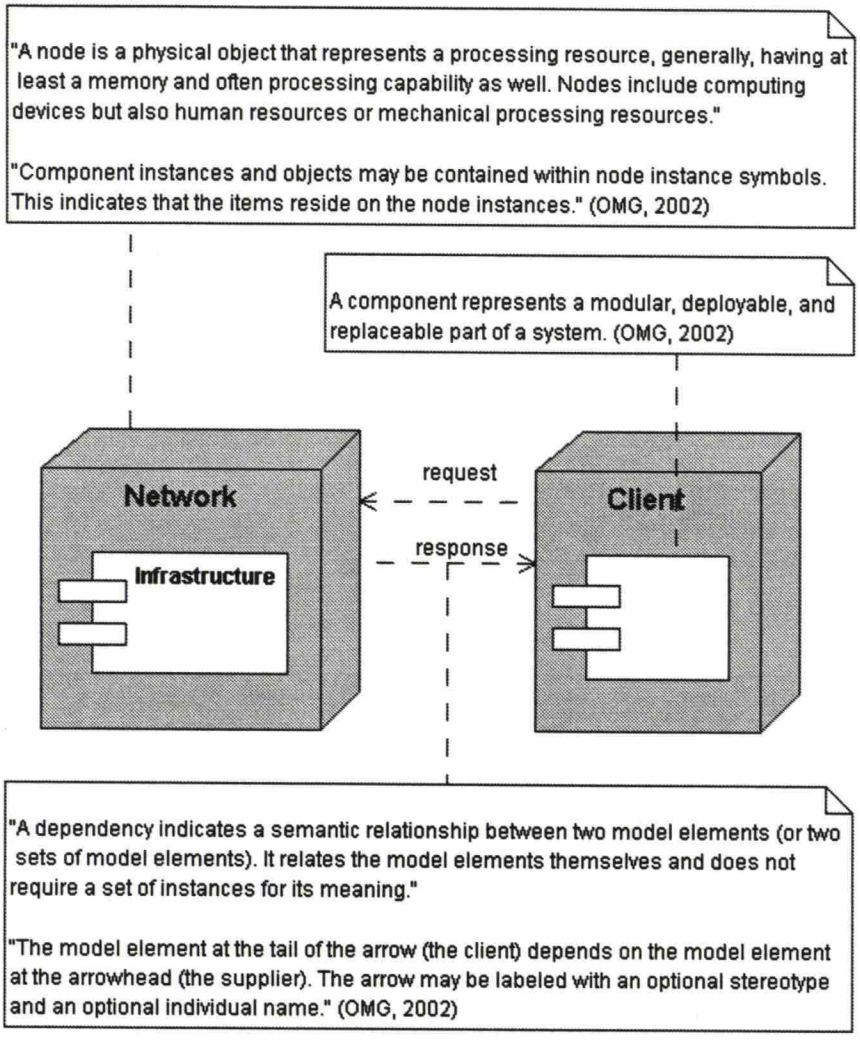
Domain model:

"Class diagram - A diagram that shows a collection of declarative (static) model elements, such as classes, types, and their contents and relationships." (OMG, 2002)



Context diagram:

"A deployment diagram is a graph of nodes connected by communication associations. Nodes may contain component instances. This indicates that the component runs or executes on the node. Deployment diagrams show the configuration of run-time processing elements and the software components, processes, and objects that execute on them." (OMG, 2002)



Statechart diagrams:

"Statechart diagrams represent the behavior of entities capable of dynamic behavior by specifying its response to the receipt of event instances. Typically, it is used for describing the behavior of class instances, but statecharts may also describe the behavior of other entities such as use-cases, actors, subsystems, operations, or methods." (OMG, 2002)